2012.04

PROGRAMMER 程序员



17 PyCon US 2012
感受开源运动的魅力

20 大数据里的小利器

58 与未知同行
敏捷开发中的反馈与反复

69 《囧记单词》创始人欧阳丹
微创新的力量

72 20小时
开发一款应用

80 从架构角度看
移动App之争

85 Siri详解
输入系统和活跃本体

90 大话推荐系统评测

100 云计算基础设施解析

114 前端框架利器
Twitter Bootstrap

118 使用 BackboneJS 搭建 SPA

135 硅谷创业之父
Paul Graham

ISSN 1672-3252

9 771672 325128 04

邮发代号：2-665　定价：15元

www.programmer.com.cn　www.csdn.net

Contents



P17

资讯

6 新闻

8 新产品新工具

10 外刊速递

13 程序天下事

17 感受开源运动的魅力
——PyCon US 2012大会见闻

20 大数据里的小利器
——Strata 2012大会见闻

22 强大的密码胜于利剑
——RSA 2012参会心得

封面报道：创业

对于互联网从业者而言，创业意味着你将经历一次自我修行，全力地追逐梦想，并有机会创造财富甚至改变世界。同时，也意味着你和你的团队将独自应对所有始料未及的挑战。本期封面报道，我们将从创业方向、创业伙伴、技术选型、融资等几个角度，探讨互联网创业的方方面面，以求让你的创业之路，不再孤单。



P26

27 正在改变世界的硅谷创业趋势

30 如何挑选联合创始人

32 创业公司如何确定技术路线

34 注重实效的初创公司技术之路

37 技术人员创业与融资的八大误区

39 “解决遗憾”是极好的创业契机

41 定义中国轻博客
——点点网创始人许朝军专访

44 精实起步
——糗事百科创始人王坚专访

47 LBS，在探索中前行
——《街旁》CEO刘大卫专访

50 最值得关注的八家创业公司

52 中国最值得关注的IT从业者创业图

管理

54 产品原则和产品评审团
Marty Cagan是享有世界声誉的产品管理专家，曾经担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他回顾自己二十多年来从事软件产品管理工作的总结和经验分享，描述了产品开发需要遵循的产品原则以及成立产品评审团的必要性。



58 与未知同行
——论敏捷开发中的反馈与反复

62 应用程序设计语言的复杂性
可以用四个关键词来区分开发规模的不同等级，分别是功能（Function）、程序（Program）、应用（Application）和系统（System）。其中，Application这个等级的开发活动被称为应用程序设计。本文是对这一规模下的语言的反思。

66 一分钟先生：创业公司如何实施股权激励
由于启动资金有限，很多创业公司都想通过实施股权激励方案，来吸引人才、稳定团队。但如何实施才能更有效呢？且听本期三位嘉宾将他们的实践经验娓娓道来。

移动

69 “微创新”的力量
——《囧记单词》创始人欧阳丹专访
因为对教育的热情，欧阳丹与朋友们共同组建了“超有爱”团队，打造了拥有百万用户的应用《囧记单词》。欧阳丹说，《囧记单词》在英语教育上的创新来自不断的学习和积累。

72 如何在20小时内开发一款应用
在本文中，作者将介绍由三人组成的团队如何在20个小时内开发一款iPhone应用程序和一款Android应用程序，这两者均由Node.js提供后端支持。

74 水果忍者
——基于返璞理念和前瞻营销的终极荣耀
尽管拥有好的创意无疑是占领市场的首要条件，但返璞归真的游戏设计理念和前瞻性的营销策略，是《水果忍者》能够成功的两个不可忽视的因素。

77 从Enyo看移动Web应用框架设计
Enyo是一个来自Palm的Aries项目的JavaScript开发框架，它所关注的方向是一般的程序框架所关注的封装性、复用性和可维护性。



80 从架构角度看移动App之争
Native App和Web App孰强孰弱？Web App是否在将来能取代Native App？这一直是移动互联网从业者讨论的话题，本文将从架构的角度来进行分析。

82 Chrome for Android对Web App发展的影响
新的JavaScript引擎、硬件加速、HTML5 Canvas的性能提升以及新的网络协议，将使Chrome内核成为Android平台上的主流浏览器内核。

云计算

85 Siri详解之输入系统和活跃本体
Siri是iPhone 4S内置的智能语音识别系统，吸引了许多用户的关注。本文将从技术层面详解Siri，主要讲述输入系统和活跃本体两部分内容。

90 大话推荐系统评测
推荐系统作为一种信息过滤的重要手段，在解决信息超载问题上非常有潜力。而怎样评测推荐系统的好坏难之又难。本文中给出了多项评测操作系统的指标，并介绍了其含义和获取方法。

2012年04月刊　总第234期

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行总编：刘江　Executive Editor-in-chief：Liu Jiang
执行主编：孟迎霞　Managing Editor：Meng Yingxia
编辑部主任：董世晓　Director：Dong Shixiao
责任编辑：陈博　杨爽　卢鹄翔
Editors：Chen Bo　Yang Shuang　Lu Dongxiang
特邀编辑：蔡学镛　池建强　冯大辉　高博　肖新光　杨栋　余晟　周爱民
Contributing Editors：Cai Xueyong　Chi Jianqiang　Feng Dahui　Gao Bo
Xiao Xinguang　Yang Dong　Yu Sheng　Zhou Aimin
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：朱凯　Art Editor：Zhu Kai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
E-mail：editor@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
E-mail：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
E-mail：market@csdn.net

发行部
Distribution Dept.010-64351431
E-mail：sales@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Zhongrun Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

本刊文章版权所有 未经许可不得转载
发现装订错误或缺页，请将杂志寄回本刊读者服务部调换

95 Hypertable应用实践
——基于BigTable模型优化业务
在2月刊《程序员》刊登的《Hypertable应用实践：比肩HBase》一文中，作者主要从Hypertable的系统层面介绍了如何改进和优化系统来更好地支持业务。本文则主要从业务层面介绍如何基于BigTable模型设计表格及导入/查询策略。

100 云计算基础设施解析
借力云平台，很多公司在业务上取得了成功，但由于云平台上开发经验的欠缺，开发者在云平台的使用上存在不少误区。针对这一现状，本文详细讲述了使用EC2、EBS和S3三个云平台产品过程的常见问题，以及如何在云平台上架构稳定可用的服务。

技术

104 C++ AMP异构并行编程解析
C++ AMP（Accelerated Massive Parallelism）是微软于2011年6月推出的异构并行编程框架，在2012年2月的GoingNative大会上，微软正式发布了它的开放规范，并在开发工具中正式提供了对它的支持。

107 Windows 8的Metro应用（中）

111 Windows 8应用开发概览

114 Twitter Bootstrap：前端框架利器
Web前端开发者每天都与HTML、CSS、JavaScript打交道，然而不少人都是周而复始地写模板、样式和交互效果，并没有想过如何将这些重复的工作整合在一起。Twitter推出的Bootstrap能够帮助Web前端开发者摆脱这种重复劳动。

118 使用BackboneJS搭建SPA
BackboneJS自从2010年发布以来，受到了业界的广泛关注。“豆瓣说”和“豆瓣阅读（阅读器）”是两款以它为基础框架搭建的SPA，本文将结合这两款产品，向读者介绍BackboneJS的方方面面。

百味

126 新书上架

128 半导体的丰收（下）
随着CPU与GPU合并成技术发展的趋势，苹果开发出了OpenCL框架，能够进行高速并行处理的能力使OpenCL成为了业界标准，被广泛应用。

132 从工程师到管理者：自然而然的转型
——雅虎北京全球研发中心高级架构师梁启鸿专访

135 硅谷创业之父：Paul Graham

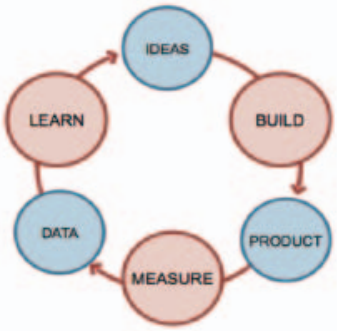
136 程序幽默

企业专栏

24 跨平台游戏引擎Unity3D初体验

创业

“创业”这个词，常常让人联想到车库、彻夜不眠、震惊世人的产品和一夜巨富故事。然而Eric Ries在2008年提出的Lean Startup则给这些关于创业、企业家和企业家精神的都市神话画上了句号。在他看来，创业是一件扎实的工作，是一项可以学习和练习的技术，而不仅仅是灵光一闪和孤注一掷。Ries的创业技术由一组源自丰田的精益方法作为指导，提倡优先采用免费或者开源软件，以用户为中心快速迭代，通过敏捷流程节约开支并尽快确定商业模式。



关于“如何创业”的问题，Ries给出的关键词是“Build-Measure-Learn”。以往所有的创业者把关注的重点放在“Build”上，但缺少了创新度量和经过验证的学习，于是无法知道一个猜想是否正确，更没有办法知道基于这个猜想正在“Build”的东西是否正确，创业就会退化成赌博。

随着《The Lean Startup》的出版，这套创业方法从2011年起得到了越来越多人的关注和认可。

安全

Duqu是一种复杂的木马，有迹象表明其制造者与臭名昭著的Stuxnet蠕虫创建者为同一人。与Stuxnet造成工业破坏的目的不同，Duqu被用来收集与攻击目标有关的各种情报。

Stuxnet最早于2010年被发现，其后感染并破坏了伊朗纳坦兹的核设施，并使布什尔核电站推迟启动。赛门铁克安全响应中心主任Kevin Hogan指出，在伊朗约60%的个人电脑被它感染。而卡巴斯基实验室则认为这类病毒是一种十分有效并且可怕的网络武器原型，除非有国家和政府的支持与协助，否则很难发动如此规模的攻击。

从其Duqu发动的攻击情况来看，它似乎只对收集密码、抓取桌面截图（暗中监视用户的操作）、盗取各类文件感兴趣。利用这种手段，网络罪犯将有足够的能力进行工业间谍活动，甚至绑架与勒索。

卡巴斯基实验室表示，Duqu部分代码可能是用一种全新编程语言开发的，既不是C++也不是Objective C、Java、Python、Ada、Lua等已知语言，只能辨认出是面向对象语言。

反病毒专家希望认识它的人能挺身而出，帮助他们分析Duqu，揭露程序的幕后创建者。这一举措果然收到成效，专家已经确认木马是由一种少见的面向对象C方言编写，并在Visual C++ 2008下编译，这种编程风格常见于十年前的商业软件，是老手所为。

移动

3月14日，软件游戏猎手发布了《2011年中国区App Store数据报告》。数据显示，2011年的App应用市场可谓“牛市狂奔”，无论是应用数量还是开发者数量都呈现出爆炸式的飞速增长。

截至2011年12月31日，中国区App Store实际在线应用为503,138款，相比2010年增长了202%。在类别分布上，游戏、娱乐、图书、教育和生活位居TOP5，合计比例超过了55%。但对比2010年度新增App发现，原本排名第一的“图书类”成了唯一一个负增长类别，其下滑比例高达20%，占总数的7.4%，被游戏、教育、娱乐和生活类反超。



付费App依旧占主导，占总数的57.3%，但与2010年相比，免费应用增长是付费应用增长的2倍之多，数据表明，开发者的盈利模式正从单一的App收费转向“免费下载+广告+道具付费”等多样化的增值方式。

“我确信会有这么一天，每个人都会开始收集有关自己的各种数据。”

——Mathematica和Wolfram|Alpha创始人Stephen Wolfram从1989年就开始收集个人数据，包括每一封邮件、每个日历中的活动、每一次键盘输入、每一次电话、房间的出入情况等各种信息。他说这些数据不单为回忆过去增加了一个新维度，对他的工作和生活也都有着重要的启发。

“巨屏手机相当愚蠢。”

——2012世界移动通信大会上，诺基亚宣布了采用4.3英寸屏幕的Nokia Lumia 900，在被问及诺基亚是否会推出更大屏幕的手机时，负责品牌推广的全球营销副总裁Niklas Savander表示，如果手机屏幕变得与平板的屏幕一般大小，不如直接推出平板电脑。

“Posterous的工程师、产品经理以及其他的团队成员将会加入到我们的团队，这将会使Twitter的服务质量得到提升，能够更好地为用户服务。”

——轻博客服务公司Posterous日前通过其官方博客宣布，公司已经被Twitter收购。但Twitter在这次收购中在乎的并非是该公司的产品Posterous，而是这款产品的开发团队。收购之后，现有Posterous团队的大部分工程师及产品经理等工作人员都将加盟Twitter。

“iOS版iPhoto已经开始使用苹果自己的地图服务，这些地图信息采用了OpenStreetMap的数据库。”

——近三年，苹果公司曾收购几家地图服务类公司，分别是2009年收购的地图厂商PlaceBase、2010年收购类似Google地球的软件Poly9 Globe，以及2011年收购3D地图技术公司C3。面对这一系列收购行为，业界普遍认为苹果将会推出自己的地图服务。

“我们想使公司成为像Google那样工程师能够晋升无阻，而且IT和HR政策都以工程师为中心的地方。我加入Reddit是因为我喜爱它。”

——原Facebook工程总监黄易山（Yishan Wong）出任Reddit CEO，并在Reddit发表博文，讲述其加入Reddit的经过与公司未来发展动向。黄易山称Reddit注重技术和工程师，不是传统类型的公司。

“即使iOS 5已经发布多时，但基于iOS 4的软件仍然非常安全。对许多开发者来说，因为他们都向前看，而无需像Android那样向后看。”

——Instapaper开发者Marco Arment在谈到程序员为什么偏爱在iOS平台开发软件时如是说。目前只有小部分Android设备在使用最新版的操作系统，对开发者来说很难开发适合所有Android设备的应用，尤其是那些资源有限的开发者。

CSDN十大资讯

2012年2月26日–3月26日

01 微软发布Windows 8消费者预览版

作者 / 王晖

巴塞罗那世界移动通信大会上，微软发布了其下一代操作系统Windows 8消费者预览版，与此同时，Windows应用商店也正式开放。微软称，尽管Windows 7是微软历史上最成功的操作系统，但时代变迁，移动终端大潮来袭，微软也必须顺应潮流，做出适合移动终端和PC的操作系统。

02 苹果发布新一代iPad

作者 / 张勇

北京时间2012年3月8日凌晨，苹果在旧金山召开发布会，发布新一代iPad，售价和iPad 2一致。新一代iPad采用Retina显示屏幕，配备了A5X处理器，苹果称其性能比现在Android平板里最强大的Tegra 3处理器的图像处理器速度还要快4倍。这还是苹果第一款支持4G LTE网络的设备。

03 白班程序员 VS 夜猫子程序员

作者 / MitchDenny

MitchDenny曾经发表博文将白班程序员与夜猫子程序员进行对比。最近文章引发了激烈讨论。他认为白班程序员通常被人领导，不在家使用开发工具，不会参与开发社区工作，把编程看成“一份工作”来对待，而夜猫子程序员往往与此相反，对编程工作更积极。

04 JavaScript排行超越Perl和Python

作者 / TIOBE

尽管JavaScript最初是给非程序员设计的脚本语言，但它灵活和丰富的特性为它带来了越来越多的应用场景。如今大多数网站越来越依赖JavaScript进行开发，在3月的TIOBE排行榜中，JavaScript的发展不容小觑，占据榜单第8位，超越了Perl和Python。

05 微软、苹果和Google三巨头的收入构成

作者 / 壹号

作为传统软件公司，微软近些年却在娱乐消费设备中赚得满钵。苹果的三分之二的收入均来自于硬件部门，而未来则很有可能依靠iOS生态系统，坐享与开发者分成。Google苦心经营的Android并没有成为其主要收入来源，广告收入的比例高达90%以上。

06 全球50大创新企业 腾讯挤进前十

作者 / fastcompany

作为与《财富》和《商业周刊》齐名的商业杂志，《Fast Company》近日评出了 2012 年全球 50 大创新企业。苹果当之无愧荣登榜首，Facebook 紧跟其后，Google 位居第三，Amazon 排名第四。而最值得注意的是，国内企业腾讯杀出重围，排在第八位。



07 雅虎起诉Facebook侵权

作者 / 张勇

雅虎3月13日提起诉讼，指控Facebook在社交网络模型、热门新闻、广告方法以及隐私设置方面侵犯了雅虎专利。分析人士称，如果有什么算是最大胆、最愚蠢赌博的话，那么这或许称得上是一件。对科技公司而言，获得专利技术主要是用来防御可能遭到的专利诉讼，而不是主动进攻。

08 企业如何招聘到高质量的程序员

作者 / Jeff Atwood

Stack Overflow创始人Jeff Atwood发了一篇文章《How to Hire a Programmer》，建议首先对应聘者进行一系列简单的在线编程能力测试，其次公司主管要关注应聘者在互联网上的个人资料。雇用适合企业文化的人很重要，如可能，请应聘者参与实际项目，看看他的实际表现。

09 黑客五分钟攻陷Chrome

作者 / samzenpus

2011年CanSecWest的Pwn2Own黑客挑战赛中，Chrome是唯一没有被攻破的浏览器。2012年法国黑客团队Vupen利用Chrome的两个安全漏洞，在比赛刚开始五分钟就成功击败了Chrome的双重安全保护机制，该团队赢得了6万美元的奖金，其中一部分将由Google支付。

10 Google年内整合旗下游戏平台

作者 / 小贝

旧金山的游戏开发者大会上，Google+产品经理Punit Soni承诺：“到明年，我们将不会再讨论Google+游戏、Chrome商店游戏或Android游戏，而是统一为Google Games。”如果Google做到这一点，或许有可能在社交游戏领域向Facebook发起挑战。

2011年最受尊敬的10大科技企业

美国《财富》3月评选出了2011年最受尊敬的科技企业，苹果雄踞榜首，IBM位居次席，Google、Amazon、eBay也榜上有名。

01 苹果

尽管创始人史蒂夫·乔布斯去世，但苹果仍在2011年取得了重大成功。在新任CEO的带领下，苹果2011财年营收攀升至1080亿美元，这主要是受iPhone和iPad销售火爆的推动。

02 IBM

1911年，IBM的前身计算制表计时公司宣告成立，它由三家公司合并而成。2011年IBM迎来了百年华诞，当年利润达到创纪录的159亿美元，同比增长7%。

03 Google

对于Google而言，2011年是重要的转型之年。埃里克·施密特2011年4月将公司大权交给联合创始人拉里·佩奇，改任执行董事长。佩奇随后对公司业务进行重组。

04 Amazon

与其他一些科技行业巨头不同，Amazon从来不会为了眼前利益而放弃自身的长远规划，这也是它2011年营收增长至480亿美元，而净利润下滑55%的原因。

05 EMC

正如EMC所看到的行业趋势，数据来源和现有数据量都有望实现巨大的增长。受益于各类硬件和软件存储系统，以及云解决方案的涌现。EMC 2011年第四季度营收增长62%，接近31亿美元。

06 Intuit

Intuit以TurboTax和QuickBooks等产品著称，主要为中小企业和个人用户提供商业与金融管理服务，其33%的营收来自于消费税产品。

07 Adobe

Adobe在2011年完成了重要的战略调整，不再拘泥于自家Flash产品，转而支持快速增长的HTML5网页标准，推出了HTML5网页动画工具Edge。

08 赛门铁克

作为世界上最大的安全软件厂商，赛门铁克在2010年一度陷入低谷，但2011年得益于移动和云计算领域的突出表现业绩而出现反弹，实现营收62亿美元。

09 eBay

eBay已经成了全球最大的电子商务和在线支付公司之一。2011年利润达32亿美元，这主要受益于PayPal为其带来的大量用户。

10 微软

微软2011财年运营利润达271.61亿美元，相比2011年的240.98亿美元，增长了13%。在经过几年的沉闷之后，我们有可能在2012年见证微软从移动领域发力。

Mozilla宣布B2G手机系统



Mozilla在巴塞罗那举行的世界移动通信大会上正式对外宣布了Boot-to-Gecko（B2G）项目，并在刷新固件三星Galaxy S II设备上展示了雏形。B2G基于HTML5和WebAPI，使用源自 Android 的内核、驱动和部分底层库，从而保证设备兼容性。系统移除了Android的Dalvik虚拟机架构，于是不再需要Java，避免了与Oracle的专利纠纷。融合ChromeOS的NativeClient技术，实现了名为Gaia的用户界面。目前系统初步实现了浏览器、设置程序、媒体播放器、应用商店、相机和电子阅读等功能。

微内核操作系统MINIX 3.2发布

Linus和Tanenbaum关于微内核操作系统的论战是20世纪90年代最著名的互联网事件之一。虽然Tanenbaum始终认为微内核是更先进技术的代表，但我们目前能接触到的真正的微内核系统并不多，MINIX可能是距离普通人最近的一个。MINIX 3.2.0增加了不少新特性，其中很多来自NetBSD仓库，比如替代了libc的用户态程序和启动引导程序。UNIX类软件可以更容易地移植到MINIX 3.2.0，Clang和ELF分别作为默认的编译器和二进制文件格式，也方便在不远的将来可以把MINIX移植到其他硬件平台上，现在MINIX社区已经在尝试移植到ARM。

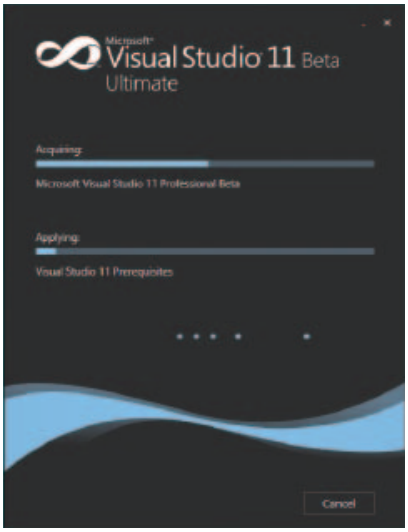


实验性地支持SMP也是新版本的增加的功能，MINIX 3.2.0的代码也与时俱进地转移到了git，这让黑客们可以更方便地hacking。

微软发布Visual Studio 11 Beta

随着Windows 8消费者预览版的发布，微软公司同时推出了开发工具Visual Studio 11 Beta和.NET Framework 4.5 Beta。作为微软最新的集成开发环境，Visual Studio 11 Beta新增Windows 8开发工具，包括以触摸为主的Metro风格界面，还有与之相关的WinRT API。开发Metro风格应用是Visual Studio 11 Beta的主推功能，Visual Studio 11 Beta可以帮助开发人员在应用开发过程中快速设计Metro风格和架构，其内置的Blend设计工具方便用户在设计过程中同步运行和查看。

为了应对日益复杂的软件执行环境，Visual Studio 11 Beta还为开发与测试团队提供了质量追踪工具IntelliTrace。



Sencha Touch 2 发布



Sencha Touch是专为开发移动设备应用而设计的JavaScript框架。通过Sencha Touch，开发者可以创建非常接近Native App的Web App，用户界面组件和数据管理全部基于HTML5和CSS3，全面兼容Android和iOS平台。

新版Sencha Touch包括更加易用的API，增强的MVC，丰富的文档。为了利用本地硬件和系统服务，Sencha Touch 2提供了大量的原生设备API接口，允许HTML5开发者利用硬件特性开发程序。同时Sencha Touch 2提供了一个应用打包工具可方便地将App直接发布到Android Market和苹果App Store。

与Twitter Bootstrap完美结合的Web图标项目Font Awesome

尽管不少开发者坚持在Web项目中使用个性化的图标，但对用户来说，一致的界面能够减少学习负担，更快地适应网站的操作。拥有这样一套高质量图标也能让开发者将精力更多地投入到应用功能本身，而不是将大量时间消耗在基础UI的设计。Font Awesome就是专为搭配Twitter前端开源工具包Bootstrap而设计的图标项目。

该项目目前已经完成超过150种高质量的矢量常用图标，具有良好的兼容性使开发者能够通过CSS控制其颜色、大小、阴影等特性。宽松的授权方式使之能够应用于商业项目。



6年磨一剑Audacity2.0.0发布



经过6年时间14次更新，开源跨平台音频编辑软件Audacity终于离开了1.3分支，步入2.0时代。

新的2.0.0版本带来了众多新功能，强化了大量音频特效，特别是噪音移除、均衡器和均一化，在音轨菜单中增加了同步音轨锁，支持在一个音轨中增加多个片段，增加了许多便于使用的快捷键，为音频输入和输出增加了新的设备工具栏，支持倒计时录音功能，对于WAV/AIFF支持从音频源快速按需导入功能。此外这个版本能完善支持FLAC音频格式，增加了自动恢复功能，在程序意外程序退出后可以恢复到崩溃前的状态。

中文Web应用框架ThinkPHP 3.0

ThinkPHP创立于2006年初，是一个开源的、快速简单的面向对象的轻量级PHP开发框架，遵循Apache2开源协议发布。ThinkPHP的目标是实现敏捷Web应用和简化企业应用开发。3.0版本主要针对底层架构进行了调整和更新，新特性主要包括改进的项目编译，增强的调试模式，增强的路由机制，提升模板引擎性能，支持RESTFul、MongoDB。它还使用了全新的CBD（核心+行为+驱动）架构模式，实现了DIY框架和类AOP编程体验。同时，3.0版本还开始提供对包括新浪SAE在内的云平台扩展支持。

Firebug 1.10a5发布，转向GitHub

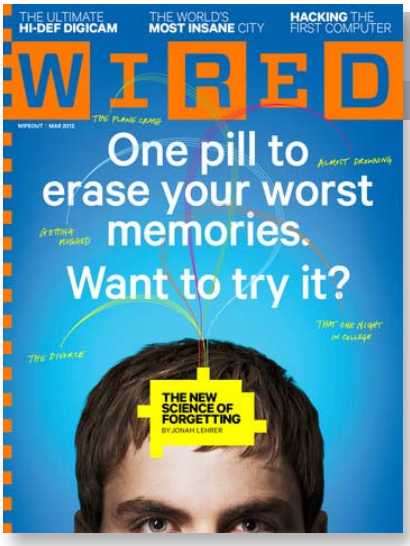
Firebug是网页浏览器Mozilla Firefox的一个扩展，开发者可以利用它除错、编辑、删改任何网站的CSS、HTML、DOM、JavaScript代码。此外，Firebug也提供Google Chrome的版本“Firebug Lite”。Firebug由Joe Hewitt撰写，他是Firefox创始者之一。Firebug还提供扩展框架支持，例如雅虎的网页速度优化建议工具YSlow、FireCookie、FirePHP等。

新版本的改进包括断点条件编辑器，支持自动完成，可通过上/下键来修改颜色，同时支持RGBS、HEX、HSLA等多种格式。另外，随着新版的发布，Firebug已将整个源码库移到了GitHub。



Python 3.3首个Alpha版本发布

该版本除了包含针对3.x分支的一系列改进之外，还使应用程序在2.x与3.x之间迁移更加容易。Python 3.3的主要改进包括：委派子生成器语法“yield from”，灵活的字符串表示形式，抑制异常上下文，重写了OS和I/O的异常层级，新增“packaging”模块，可构建“distribute”和“distutils2”项目，新增“lzma”模块以支持LZMA/XZ压缩，现在函数和类有一个新的“qualname”属性以表示从模块顶层到定义的路径。此外还新增“faulthandler”模块，用以帮助诊断崩溃问题。



Wired
2012.3

黑客马拉松大赛如火如荼：构想与编制下一个超酷应用

厨房里堆满方便食品和提神饮料，工作台上有人在挑灯夜战，有人因为累极而进入梦乡……不同团队连续48小时的编程鏖战，目标是各自交出能够正常使用的Android应用程序——此情此景所呈现的是某一场黑客马拉松大赛（hackathon）的“执行概念”阶段。在此之前，参赛团队已经走过了“创意闪现”和“成立团队”阶段，此后将迎来“成品报告”和“接受裁判”步骤，最后一个步骤——“收获奖金”则只有胜出者才能幸福地品味。这样的黑客马拉松大赛2011年在美国举办了200多场。

黑客马拉松兴起于1960年代，当时尚属程序员们自发竞赛。直到1990年代，时下流行的黑客马拉松大赛模式才浮出水面。参与和关注黑客马拉松大赛的机构和个人可谓各怀心事：财富500强公司希望通过黑客马拉松提高产品认知度，AT&T、微软、诺基亚乃至联合利华等大玩家都通过举办过黑客马拉松获取创意；风险投资人希望通过黑客马拉松找到值得投资的新面孔和新点子；技术猎头希望通过黑客马拉松找到技术专才，增加收取佣金的机会；程序员本人则希望引起风投的关注，获得资金，大展事业宏图。

黑客马拉松比赛在美国发展得如火如荼，而更为简单易用的软件编写工具、各家公司愿意让第三方分享自家数据的新颖意愿为这燃烧的野火再送清风，从而炼出层出不穷的创意和业务。仅以苹果公司iOS为例：自从2007年首次开赛以来，已收获50万项应用程序，为开发者们产生了30亿美元的业绩（Android的40万应用程序收入约1亿美元）。

尽管开发应用程序和赚钱的机会出现爆炸式增长，编程所需的时间和预算却日益接近于零。程序员们通过剪切和粘贴几行代码即可将核心功能添加到应用程序中。也就是说，现在是进入应用程序业务的最佳时机。借这个时机获得风险投资并大获成功的程序员不胜枚举，被Skype收购的GroupMe就是其中一例。

忘忧药，永远抹去痛苦的回忆

自从古希腊时代起，人们就将记忆想象成一种长久、可靠存在的稳定形式。但科学家在过去十年中却意识到，我们的记忆并非凝滞不变的数据仓库。虽然每一次记忆给人的感觉都仿佛一次忠实的事实陈述，但这种真实感却是天大的谎言。新的研究表明，每当我们回忆起一桩事件，大脑中的记忆结构就会随着当前感受和认知发生歪曲。记忆并不是一旦形成就一成不变，而是每次想起都会进行重组。

这种新的记忆模型并非纸上谈兵，神经系统科学家当下已给出了关于记忆变化的原因和过程的分子学解释。大量论文证明，回忆活动需要合成某种蛋白质，这与记忆最初产生时对化学物质的需求异曲同工。如果大脑想要保留什么东西，所要依靠的不过是为数不多的几种化学物质，科学家已经识别出一种与记忆有关的化合物——PKMzeta。为了抹去大脑中不想要的历史，可以依靠的仍然是为数不多的几种化合物，例如ZIP（Zeta-Interacting Protein）即是一种已经出现的PKMzeta抑制剂。目前，研究者必须将记忆药剂直接注射到啮齿目动物的大脑中，而未来的治疗则会使用目标明确的抑制剂，如ZIP的改良型，这种抑制剂仅对皮层的特定部位有效，并且正好在回忆出现的那一时刻有效。

神经系统科学家对记忆的定义十分宽泛，不仅包括孩提旧事，还包括长期存在的各种精神困扰，如创伤后精神压力症以及癔疾，甚至还包括神经病之类的疼痛障碍。抹除痛苦记忆的忘忧药因此大有用武之地。将来会出现一系列的药物，用于抹去各种各样的记忆——旧情人的香气、分手后的痛心疾首……无关记忆则完好保存。

现在的问题是，我们真的有必要放弃吃一堑长一智的机会抹除痛苦的记忆吗？我们能够不滥用这种记忆编辑技术吗？实验室里的小老鼠固然不会开口抗争，实验室以外的伦理学家和临床医生又会怎么说呢？

专访黑客历史学家 George Dyson

20世纪最具威力的两项发明——核弹和计算机出自同一时代、同一群年青人。可是，与大名鼎鼎的曼哈顿计划

（第二次世界大战中美国原子弹研究计划）相比，计算机的起源显得默默无闻。出身计算机世家的历史学家George Dyson在其新书《图灵大教堂》（Turing's Cathedral）中讲述了阿兰·图灵、约翰·冯·诺依曼等一帮子天才小子创造计算机及预见计算机未来盛况的轶事。Dyson向《连线》前主编凯文·凯利（以下分别简称Dayson和KK）讲述了“数字宇宙”的“大爆炸”。

KK：我们在进行电话访谈，这时我们是否处于这个数字宇宙中？

Dayson：当然。您正在用数字录音机将这次对话录制到一个位于个某芯片上的空地址矩阵中，写入速度为每秒44千字节。这个写满数字的地址空间就是数字宇宙。

KK：这个平行光速宇宙是如何开始的？

Dayson：这个广袤无比的数字世界几乎能让我们随时获取任何内容，它可以追溯到MANIAC中的第一个地址存储器，大小仅有32×32×40字节，即5千字节，或者说这个空间可以录制这次谈话的几分之一秒的内容！

KK：这个宇宙的膨胀速度有多快？

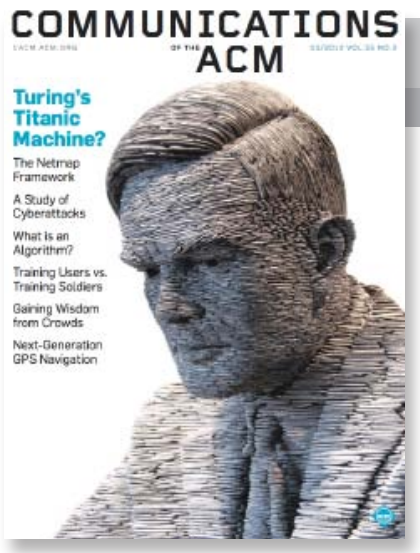
Dayson：正如宇宙形成之初，爆炸多过膨胀。我们身在其中，难以察觉。我曾经测量过，数字宇宙在存储端的膨胀速度为5兆字节每秒，处理器的膨胀速度为每秒2兆晶体管。

KK：在创造传奇的过程中，有哪些英雄人物？

Dayson：阿兰·图灵是原始创意逻辑学家；朱利安·毕格罗（Julian Bigelow）是造出实际机器的工程师；生物学家尼尔斯·巴黎塞利（Nils Barricelli）监督发展情况；约翰·冯·诺依曼赢得政府资金和信心，促使项目启动；约翰·冯·诺依曼的妻子克拉利·冯·诺依曼（Klari von Neumann）写代码。

KK：这个数字宇宙奔向何方？

Dayson：我们创造了这个不断膨胀的计算宇宙，它敞开怀抱迎接所有事物的发展演变。它越转越快，规模远远不止每年翻一倍。就算有Google和YouTube以及Facebook帮忙，我们也无法将它耗尽。我们无法真正了解这个巨大无比的空间中装着些什么。在人类看来，计算机99%的时间都在空转，只知道等待下一个指令。虽然计算机在等着我们提供指令，但越来越多的计算却脱离我们而发生，因为计算机会为计算机写指令。正如图灵用数学方式展示出来的那样，这个空间无法监督。随着数字宇宙的膨胀，这桀骜不驯的一面也会如此。



Communications of ACM

2012.3

众人拾柴火焰高 —— 在线游戏玩家通力破解科学难题

为了弄清帮助人类免疫缺陷病毒（HIV）进行繁殖的一种蛋白质的分子结构，美国华盛顿大学的科学家们已经摸索了十多年。了解这种病毒的形态有助于科学家们研制攻克病毒的药物，可科学家们一直无法辨认出这种病毒。于是，他们向开发出 Foldit 在线游戏的同事们求助，这款在线游戏给玩家的题目是对蛋白质进行重排，使蛋白质达到能量最低形态（最有可能在自然界中存在的形态）。

游戏吸引了 57,000 多名玩家，大部分人没有接受过任何分子生物学方面的训练。然而不出三周，他们得出了答案，随后发表在《自然——结构与分子生物学》杂志上。研究人员说，这可能是第一个通过在线游戏解决的长期科学难题。

游戏创造者声称这款游戏让人们看到了未来的惊人一例。“对于超级复杂的问题，非专业人士大型群体可能好过计算机……我想本来没有人相信人类会高明这么多。”

类似的例子还有一下几个。

■ 麦吉尔大学的两位助理教授设计了Phylo游戏，帮助他们对齐DNA序列用以研究遗传病。自从2010年底开始使用Phylo以来，已有35,000人玩过这个游戏，

游戏中的基因对齐程度提高了约70%。

- 卡内-基梅隆大学一名副教授开发了ESP游戏，玩家在分辨游戏图片的同时，可以为图片创建电子跟踪标签，科学家希望计算机能对这些标签应用机器学习技术，学会自行分辨图片。

- 华盛顿大学研究人员利用Rosetta@Home游戏控制个人电脑CPU的空闲时间。

尽管计算机能够处理海量数据，未来也终将能解决所有难题，可是，目前科学界中的许多难题尚无法通过计算机解决。而通过对互联网和社交媒体善加利用，研究人员就能驾驭人类生来就有而计算机无法掌握的才能——直觉、高级视觉处理以及对现实世界的理解，应对某些难题需要这些能力。最终，人类的处理有可能帮助计算机学会如何执行这些任务。

图灵机是计算机器中的泰坦尼克吗？

由于阿兰·图灵与泰坦尼克号邮轮同在1912年诞生，图灵机被戏称为计算机器里的泰坦尼克号，它们既让人们看到科学和工程上的成就，又提醒人们自然事件存在固有的不确定性。

关于图灵计算模型是否妥当这个问题，至今难以得出定论。“常规科学”对此三缄其口，对这些问题感兴趣的人们争论不休，互有异议。

理论上的不确定性和建模上的不确定性都是原因所在。数学方面，我们基于图灵机模型及其oracle强化原型形成了成熟的经典可计算性（以及不可计算性）理论；另一方面，我们有着互联网、人工智能和自然界，它们所具有的密切相关的模型有时被称为新计算范式。

比如多年来，停机输入集以各种各样的伪装出现。可这种对象的计算是从什么意义上讲的？为什么有意义？

简短回答起来，这种对象的计算很像出现在计算机屏幕上的曼德勃罗特集。从可计算意义上看，它是可枚举的。大量研究工作都以探寻各种不可计算对象（不只是停机集的不同写法）为方向，再就是让我们所遇到的各种不可计算对象更好处理，在现实中更和谐。

一些主要观点包括：

- 无论是现代计算经验，还是我们的自然界，都充满针对其普遍有效性的挑战；

- 通过人工智能、物理和计算实践，我们应该学会给予“具体化”计算、信息及过程充分的尊重；

- 理论建模以及与计算实际相结合都是促成过去和未来进步的协作成分；

- 图灵留下的财富表明：我们大有可能阐明和把握“世界如何计算”的奥秘。

图灵的“泰坦尼克”机器大有希望避免下沉，反而在这个动荡的自然环境下得以具体化，满载着奇思妙想，带着计算结构驶向迷茫的远方。

何为算法？

2011年7月在法国举办的第14届国际逻辑学、方法论及科学哲学大会（CLMPS）上，举办了一场以“何为算法？”为主题的专题研讨会。在图灵100周年诞辰庆典到来之际，提出这样的问题似乎颇为奇怪——难道图灵没有断定这个问题的答案吗？难道答案不是“算法就是图灵机”吗？

但将图灵机和算法混为一谈却是对图灵发表于1936年的论文《论可计算数及其在判定问题上的应用》的误解，图灵的目标是定义可计算性，而不是定义算法。他在论文中称，每一种基于自然数的能由人类计算机计算的函数都同样能由图灵机进行计算。他并没有在论文中说图灵机给出了一种常规算法模型。因此，CLMPS大会主题研讨会上提出的问题十分精彩。

令人惊讶的是，会上的两位主要发言人Y. Gurevich和Y. Moschovakis对这个基本问题给出了两个截然不同的答案。

Gurevich认为，每种算法都可以从一台抽象状态机的角度进行定义。直觉上，一台抽象状态机就是一台运行于各种状态（即任意数据结构）的机器。主要要求是机器的一个步骤只能引起状态的一个本地有界变化。这个要求与图灵机的一次一格运行以及冯·诺依曼的有界宽度运行相符。

与之相反，Moschovakis则认为算法是从递归器（recursor）角度进行定义的，递归器是一种建立在任意运算（作为基本式）上的递归描述。例如阶乘函数，可以将乘法作为基本算术运算，从而以递归形式描述；而欧几里得最大公约数算法则将余项函数作为基本运算，从而进行递归描述。

那么算法是抽象状态机还是递归器呢？人们从数学上可以看出，递归器可以是抽象状态机的模型，而抽象状态机可以是递归器的模型。那么，谁主谁次？

算法既是抽象状态机，又是递归器，仅取二者之一无法完全说明何为算法。借鉴物理学上的波粒二象性，看来，“算法二象性”才是计算科学的基本原理。

（感谢译者李芳支持）

Windows 8与 WinRT的挑战

Windows 8是我们的未来，无论你是否接受，挑战总会到来。

千呼万唤的Windows 8 Consumer Preview终于在2月29日的MWC世界移动大会上发布了。没有选择年初的CES，而是选择了移动通信领域的MWC来发布Windows下一代操作系统，这个耐人寻味的举动本身就说明——Windows 8是一个移动操作系统。

2011年9月发布的Windows 8 Developer Preview版本的确是一个只能预览的版本，而这次的Consumer Preview则已经达到半可用状态了。自从Windows XP发布，Windows用户体验已经停滞了很久。而Windows 8则颠覆了很多老用户熟悉的界面元素，比如开始按钮、标题栏和关闭按钮等。Windows的传统界面是按照鼠标交互的方式设计的，而新的Windows 8 Metro Style应用程序的交互方式完全是按照多点触摸进行设计的。

Metro Style应用程序对于Windows Phone开发者来说并不陌生。不过，Windows Phone 7上的Metro UI只能使用Silverlight技术来实现。Windows 8上的Metro Style应用程序则能够支持C#、VB.NET、C++和JavaScript三种方式，而且底层API也从Win32变成了WinRT。自从Windows 2000开始，Win32 API几乎伴随了我们这一代程序员的整个职业生涯，MFC时代熟练使用Win32 API是高手的标志。后来有了.NET，使用P/Invoke调用Win32 API也是我们解决某些棘手问题的不二法门。但WinRT取代了Win32之后，我们再也不能模仿鼠标输入事件了。

WinRT是一个Native的API库，所以C++可以进行调用，而C#/VB.NET则是通过.NET封装来调用的，界面描述使用XAML，最大的挑战则是来自于JavaScript调用。微软使用JavaScript语法来描述了Metro Style UI，但底层类库的调用一个挑战是仍旧基于WinRT。所以对于传统的JavaScript开发者来说，仍旧需要学习WinRT类库的调用方式。另外一个挑战是，使用JavaScript来编写的Metro Style应用，代码是无法复用到普通的JavaScript应用中的。这虽然符合微软的利益，但不符合开发者的利益。所以，在很长一段时间内，Windows 8上最活跃的开发者社区仍将是.NET，因为C++开发者要学习XAML，而JavaScript开发者要学习WinRT。

在编程模型上，Windows 8向Windows Phone 7借鉴了很多有益的地方，比如所有应用程序都是运行在沙箱中，有自己的独立存储空间，后台运行时进行严格限制，从这些方面来保证操作系统的安全和性能。当然，Windows 8也修正了Windows Phone 7中的一些问题，比如无法进行程序间的相互调用。Windows 8是使用Contract来解决这个问题的，Contract的机制类似于Android中的Service，为程序间的调用提供一种标准接口。首先，由被调用者声明Contract，然后由调用者进行相应的调用。应用程序可以声明一个由Search或Setting调用的Contract，在搜索系统功能中，就可以加入自己的搜索结果。

Windows 8另外一大卖点是Windows Store的引入，能够将软件送达每台PC的桌面，恐怕是每一个ISV的梦想吧。Windows Store的游戏规则与Windows Phone Marketplace类似，采用7:3分成，超过2万5千美元的，则按照8:2进行分成。不过，Windows 8面临的问题要比iOS复杂，最初的压力会来自于原有的软件分销体系，Windows Store模式的物流成本趋近于零，而软件数量趋近于无穷，这种典型的长尾理论应用会造成之前分销体系的崩溃，但分销体系的压力也许能够迫使微软作出一些妥协。目前已知的是，Windows 8仍旧允许用户自行部署应用程序，但这种方式会对Windows Store上的软件形成非常大的反盗版压力。不知道微软如何平衡这两种新旧体系同时共存的问题。

Windows 8是我们的未来，无论你是接受，还是反对，该来的挑战，总是会来的。



马宁

北京支点联游CTO，OpenXLive开发者。曾任微软亚洲工程院软件工程师，从事Windows Embedded Compact开发工作。加入微软之前，连续四年获得“微软最有价值专家”称号。

我的数据、你的产品、中间是技术壁垒

很多公司意识到数据容灾备份的重要性，但似乎还没对数据库产品提起足够重视。

如果你走进任何一家企业的数据中心，然后指着它的数据库服务器问一位DBA：“嗨，兄弟，这是你们的数据吗？”估计他会首先一愣，然后给予你很肯定的答复。但真实情况似乎并不是这样。

当我们把自己的人民币存款从一家银行取出来转存到另一家银行时，钞票还是钞票，我们可以选择转存到哪家银行或者干脆自己持有现金。同样的，当我们发觉自己的货物保存在某家仓库不太满意时，可以把货物从这家仓库转运到另一家仓库。简言之，现实生活中我们的资产委托对方保管时，资产还是这些资产，但我们的信息资产进入了数据库以后，事情就不是这样了。

当数据保存在Oracle、MySQL、SQL Server时，数据结构、索引、日志这些内部的特质就已经发生变化，同时外部访问所用的SQL语句差别也不小，事实上我们没有办法像转场货物那样方便地直接把自己的数据从一个数据库产品迁移到另一个产品。原因在于我们的数据一旦进入某个商用的、开源的或者NoSQL数据库之后，其内容已经脱离我们——数据所有者的控制，一旦未来发现数据库存产品在这样那样的问题，我们自己的数据资产将面临丢失的风险。现实中，如果我们的货物因为寄放仓库一方的原因出现霉变、腐烂、破损，我们可以向仓库保管方进行民事索赔，但如果因为数据库产品升级、系统异常等问题导致数据明明在那儿，却无法启动加载时，根据安装数据库软件时我们往往看都不看一眼的法律声明，这些责任基本都在用户一方。也就是说，如果数据丢失，找不回来我们需要自己承担责任。

但这还不算结束，随着各家厂商、数据库组织发展目标的不同，关系数据库的“方言”问题日益严重。每当数据库厂商发布新产品时，我们总能看到Oracle多了哪些PL/SQL语法特性、SQL Server增加了哪些新的T-SQL语句。对于“铁了心”用某款产品的用户而言，这是好消息，因为不用等待ANSI漫长的SQL标准审核、发布过程，自己需要的语法特性可以随着数据库产品的版本快速发布，这点是积极的。但隐患也在此，因为今天好用的数据库不代表一直好用，而且随着对某种SQL“方言”的深入使用，未来应用程序对于特定数据库产品的依赖也在不断加强。

这几年，随着Hibernate框架，以及嵌入语言内部的LINQ、JPA（Java Persistence API）的流行，似乎应用逻辑与具体数据库的依赖性趋弱，但这只是表面现象，因为这些ORM（Object-Relational Mapping）框架能提供的只是“通用”部分，对于大型应用尤其是大型企业的关键应用、大型电子商务应用而言，使用ORM意味着在用运行阶段的性能换取开发阶段的便利性。这个折中对于这种大型的数据应用并不划算。为此，很多用户需要费尽心力地针对数据的物理存储和分布式特点，将SQL“方言”用到极致，通过封装自定义的存储过程、函数、任务供ORM框架或其他管理软件使用。这样，用户只能在特定数据库产品的路上跟着厂商一直走下去。这个问题在NoSQL数据库上体现得更为明显，因为这些NoSQL数据库——No SQL，即他们还没形成产品间通用的数据查询、修改语言的国际标准，对于使用中体验不理想希望迁移产品的用户而言，需要面对的就不仅是两个相近的“方言”，而是一门全新的“外语”。

随着商务智能和数据仓库概念的流行，ETL工具似乎表面上渐渐缓解了这个问题，但这只是“表面”，因为它将存储特征和SQL“方言”（或多维数据语言）的技术壁垒从数据库部分后移到数据仓库部分。问题依然存在，只不过现在绝大多数企业和互联网公司的关注点还在忙于如何让商务智能更“敏捷”、更“智能”，并没到考虑迁移数据仓库的阶段。

当数据成为一家单位商业的核心竞争力之后，你的数据是否真的是“你的”、想用的时候是否真的“让你能用”……这些都是必须考虑的问题。尽管经过9/11和几次大海啸之后，很多公司意识到数据容灾备份的重要性，但似乎还没对自身信息资产的载体——数据库产品提起足够重视。企业信息资源，尤其是核心信息资源的载体无关性是每个企业信息决策者需要花时间想想的问题了。



王翔
软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

B2B2C技术选型与推广思考

关于自建B2C官网或者自建B2B2C平台采用什么技术开发，我的建议是采用已有的成熟的产品。

陆陆续续，一些朋友买了我2011年出版的那本《B2B2C网上商城开发指南——基于SaaS和淘宝API开放平台》。其中还有不少是传统企业的客户，他们发邮件或者QQ向我询问开发自己B2B2C平台的事情。我那本书是2010年10月开始写的，2011年4月出版的。由于技术发展太快，在这期间，我对架构的认识也发生了不小的变化，现在我把自己学到的一些最新技术，再次写出来与大家分享。

无论是具备互联网基因的新型电商，还是传统企业客户，自建一个B2C网站都不是一件小事，2012年初，我曾在《销售与市场》杂志发表了自己的观点：自建B2C其实就是搞流量，非常困难，但传统企业出于自身的考虑，一般是不会放弃自己的B2C官方网站建设的。慢慢大家也达成了这样一个共识，以淘宝或者以京东为核心，结合其他平台销售渠道与官网的B2B2C兼分销平台，将是以后强势传统品牌商的电商之道。有互联网基因的，可以从淘宝操刀；否则建议从京东操刀。下面我们就浅论自建的B2C官网采用什么技术开发。对于像修正药业、海尔、美特斯邦威、爱慕这类有数百家经销商、分销商以及自己遍布全国各地门店的企业，完全也可以建立自己的B2B2C平台。

关于自建B2C官网或者自建B2B2C平台采用什么技术开发，我的建议是采用已有的成熟产品。如果是面向国外客户的外贸型企业，建立B2C网站的最佳选择就是Magento。Magento原来是一个名叫Magento的开源电子商务平台厂商开发的，2011年6月，这家公司被eBay收购，成为eBay X.Commerce部门的一部分，X.Commerce部门的任务是将围绕eBay所说的“开放电子商务平台”打造富有活力的开发者生态系统。如果自建的B2C官网是面向国内客户的，则建议采用商派的Shopex，在它上面在做二次开发就可以了。开发能力强的，也可以采用商派的开源EcShop产品。如果你想做成一个像淘宝或者京东那样的面向很多合作伙伴那样的B2B2C平台产品，就真没有很合适的参考对象了，商派的EcMall虽然也是多店铺的，但对一个真正的多店铺的平台来讲，还是太简单了。如果真的要自己开发B2B2C平台，我建议数据库采用NoSQL数据库MongoDB，前台展示使用PHP、Java甚至Scala都可以，但也要选自己最熟悉的技术去做。Cache技术可以采用使用起来最简单，功能又非常强大的Redis。对于多店铺和后台数据录入功能的架构设计和实现，可以参考我那本书。MongoDB和Redis可以保证B2B2C网站大数据量的并发，而自建的B2C官网，由于访问量较小，采用基于MySQL数据库的像Magento、Shopex就可以满足需求。

对于B2C网站的推广，我还想多说一句的是。其实大家也都明白，自己独立的B2C官网，几乎没有人访问，如何做到提高自己的人气和流量呢？对于特有钱的能赔得起的，咱们就不多讨论了，而对于一定要自建B2C网站的企业来讲，如果有这么一个B2C产品，能够形成CPS或者CPM联盟，就非常理想。比如，麦包包2011年花了海量的钱，去做自己的域名推广，名气上去了，但由于只卖自己的包，访问量还是极少，如果麦包包也可以卖好乐买的鞋、美特斯邦威的服装，是不是就不一样了呢？而好乐买的鞋、美特斯邦威的服装并不需要麦包包自己去上货，或者自己去录入。假设麦包包、好乐买、美特斯邦威的官网，采用的是支持同一个插件技术的B2C产品，麦包包、好乐买、美特斯邦威利用这个插件，在自己的官网上开辟一个第三方接入的插件，自动显示来自于第三方B2C网站的产品，展示在自己的官方网站上，形成B2C网站联盟，是不是就可以跟淘宝和京东竞争了呢？当然，这种插件技术，并不是现有的CPS联盟插件那么简单。要能真正在一个B2C官网上支持消费者购买所有的商品，也不仅仅是一个链接那么简单。比如在麦包包下订单，购买了美特斯邦威的牛仔裤，就可以直接传输到美特斯邦威的后台，这也是一个解决B2B2C问题的思路。



邢波涛
北京新软孚信息技术有限公司技术负责人。关注SaaS管理软件和B2B、B2C电子商务的融合。

装备制造技术出版物标准 S1000D

复杂的装备制造业，越来越依赖于技术出版物来支持各类信息内容的描述。

一年一度的S1000D用户大会将于6月18日在美国丹佛召开，广泛用于航空等高端装备制造领域的技术出版物标准S1000D将是本次大会的主角。用户大会围绕S1000D将展开多层次的技术讨论和经验交流，议题包含S1000D项目实施案例分析、S1000D项目管理经验、S1000D与各类出版物格式转换，S1000D多媒体应用，以及用于S1000D编辑和管理的各类工具。

S1000D是面向技术出版物制作和发布的国际标准，使用XML结构来描述、管理和发布技术文档。S1000D最初由ASD（欧洲航空与国防工业协会）作为军用标准设计并提出，用于记录军用飞行设备的装备维护和操作信息。从S1000D第二版开始，其适用范围就开始向更广泛的装备制造领域拓展，包含陆上设备、航海装备并逐步应用于各类商用和民用设备技术文档。

S1000D由专门的管理委员会更新和维护，并负责协调各方意见达成一致。目前通用的4.0版本由ASD、AIA（美国航空工业协会）、ATA（美国航空运输协会）以及领域用户共同参与制定。其中ATA的鼎力支持为S1000D标准的民用化进程注入了不竭动力。ATA是美国历史最悠久的、规模最大的、全国性的航空公司行业协会。在民用飞机客户服务领域，ATA参与制定的文档规范在制造业领域具有很高的权威性和可操作性。在使用上，S1000D涵盖了技术文档的计划、管理、生产、交换、分发等各个环节，覆盖工程性项目完整的生命周期。符合S1000D规范的技术文档以数据模块（Data Module）形式被创建，数据模块表示在技术文档中粒度最小且能够对信息进行完整描述的内容块。数据模块包含模块标识、状态信息及一个存放技术文档信息的内容单元。数据模块描述的内容及结构体包含业务规则信息、描述性信息、故障信息、零部件数据信息、维修及检查信息、程序及过程信息、技术知识库信息等各项内容。

在信息内容上，S1000D规定了适用范围、内容组成、业务规则、规则分类以及包含各类数据模块XML Schema的内容结构、元素、属性在内的信息内容生产和交付方法。其中，S1000D的业务规则分类是其区别于其他基于XML的技术出版物标准的主要要素。S1000D业务规则分类定义了技术文档各应用层面的细节，由近千个业务规则决策点（Decision Point）组成，如产品识别码、图形符号的尺寸和缩放比率以及与相关业务标准的接口规范等内容。

S1000D业务规则分类包含十个大类。其中，一般类定义了实施S1000D的总体业务规则，包含版本号、术语定义等；产品定义类定义产品相关的数据模块和编码策略；维护与操作概念类包含产品维护级别及操作信息等；安全保密类包含密级划分、版权标记、信息使用与公开的限制、访问权限、信息销毁指令等；业务过程类描述技术出版物与综合保障、供应计划、工程设计、培训等业务的关系；数据创建类用于定义文本、图形、多媒体对象的创建规则，如术语规则、标记规则、表达规则、样式规则等，以实现技术出版物内部及技术出版物与培训资料间的信息重用最大化；数据交换类定义了厂商与用户间交换数据应遵守的规则，如数据表单要求、数据迁移协议等；数据完整性与管理类用于保证信息创建者与客户双方的数据参照完整性；历史数据转换、管理和处理类包含信息源和目标之间元素及属性的对应关系，以及技术出版物中包含历史信息的规则；数据输出类定义了S1000D数据的输出形式，包括页面出版物、交互式电子技术出版物（IETP）、多媒体出版物、SCORM等多种形式。

对于复杂的装备制造业来说，其专业化细分程度日益增强，工程协作性要求也逐步提升，越来越依赖于以计算机为基础的技术出版物来支持各类信息内容的描述。作为国际化的技术文档标准，S1000D的应用减少了高端制造业领域的技术信息维护成本、降低了数据转化为配置项的难度、提高了技术文档输出及跨平台转换的效率，有效保证技术文档的管理稳定高效。S1000D的使用，将帮助使用者在原有装备升级改造和新装备引进的过程中保证各类技术文档的有序管理，能够在降低信息生成成本的同时避免信息内容的重复生成，且更有助于制造业产品的技术描述在世界范围内维持一致性和通用性。



高昂
中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

感受开源运动的魅力

PyCon US 2012大会见闻

文 / 骆逸

PyCon是Python社区的年度大聚会，2003年始于美国华盛顿特区，由社区成员自发组织。经过9年的发展，PyCon在全球各个主要地区都有活动。PyCon的宗旨在于促进热爱Python语言的开发者面对面交流，组织培训，以及进行开源Python项目的短期开发（Sprint）。

PyCon US 2012于3月7日–15日在硅谷圣塔克拉拉会议中心举行。尽管早在2004年我就开始使用Python，但以前没有机会参加PyCon。恰逢3月要去硅谷旅行，于是早早报了名。

Python语言已在高科技和金融等行业中得到广泛应用，这次在硅谷举行的PyCon规模空前。参会注册和赞助商报名都是爆满，不得不提前截止。Google、Facebook、Microsoft、Dropbox、Evernote、Heroku、Dotcloud等数十家知名企业都是赞助商。大会开幕时，数千平米的大厅座无虚席。当大会主席Jesse Noller宣布本届大会共有2257名Python开发者参加时，顿时掌声雷动。



Jesse Noller宣布PyCon大会开始

Keynote引人入胜

本届大会邀请了知名人士Stormy Peters（来自Mozilla）、Paul Graham（著名黑客、YCombinator创办人、《黑客与画家》作者）、David Beazley（著名Python黑客，SWIG作者）和Guido Van Rossum（Python语言创始人）作主题演讲。



Paul Graham分享7个令人生畏的创业点子

作为美国新一代创业者的精神领袖，Paul Graham的演讲自然是关于创业。他认为最大的创业想法都是令人生畏的，并提出了7个看似不可能的创业点子：1. 开发新的搜索引擎，一个为黑客而建的搜索引擎；2. 重新发明Email，提高沟通效率；3. 取代大学；4. 干掉好莱坞，开发互联网影视；5. 建一个新的苹果公司；6. 让摩尔定律复活，充分发挥硬件并行计算的潜力；7. 持续诊断，不必等待生病了才去就医。最后，他还谈了创业的竞争策略：要避免正面进攻。他的演讲非常有趣，演讲稿已经发布在他的个人网站上。回头细想，他指出的一些方向已有初创公司在尝试，假以时日一定会有所突破。

David Beazley介绍了新一代Python解释器PyPy。他对PyPy的实现进行了分析：PyPy的开发者首先定义了一个Python语言的子集RPython，然后用RPython构建了一个复杂而强大的编译工具链。该工具链可以将Python Code Object（bytecode）翻译成C代码。一个原本耗时8秒的Python Fibonachi序列程序，通过PyPy翻译为C代码再执



Python创始人阐释近期语言热点

行只需0.17秒。PyPy项目非常有挑战性，开发已经进行了9年之久。David Beazley鼓励大家去琢磨PyPy，同时还提醒要小心PyPy项目的复杂度（如果把标准库函数和单元测试代码包括在内，PyPy项目代码已经超过120万行！）。

Python“教父”Guido Van Rossum的演讲则围绕近期Python语言发展中的重大争议展开。他首先指出有些争议纯属Troll，没有实质意义，比如Python和Ruby孰优孰劣。接下来他阐释了对一些热点话题的看法。其中比较有意思的几点是：Python 3没有失败，而是进展良好，unicode literals将会回来，NumPy也正被移植到Py3k；PyPy是非常有潜力的一个Python实现，但它目前并不能取代CPython（他请在生产环境中使用PyPy的人举手，基本上无人响应）；要处理动态类型带来的类型不匹配问题，不能依靠编译器检查，而是要靠测试；全局锁（GIL）经常被误解，在大多数时候它不是问题；在异步I/O和并发方面，他倾向于gevent，不喜欢回调（callback）；函数式编程是一个有新意的想法，但很难成为主流；CPython现有的引用计数模型工作得很好，不需要新的GC。他也提到了将Python置入浏览器和Android的努力，但由于相关厂商的原因，都有困难。他最后指出一门语言如果被越来越多的人使用，在语言演化方面会趋于保守。在演讲结束后，我就Py3k和Python-Android问题再次请教他。他认为Python 3.3已经是production-ready；如果项目仅依赖标准库，则现在就可以用Python 3来开发。在Android方面，他已和Android负责人沟通过，试图说服他们支持Python，但目前尚无结果。

专题培训和讲座精彩不输Keynote

在大会开始之前，PyCon组织了32个专题培训（https://us.pycon.org/2012/schedule/tutorials/），主题广泛，涉及高级Python编程、优化、数据分析、机器学习、社交网络分析、自然语言处理、Web编程、并发编程、游戏编程和GUI编程。

本次PyCon的讲座内容也非常多（https://us.pycon.org/2012/schedule/），热点包括：PyPy、软件测试、元编程、数据处理和挖掘等。下面是我个人感兴趣的一些讲座。

- Stop Mocking, Start Testing：Google Project Hosting是一个大型Python项目。来自该项目的两位工程师分享了他们如何改进项目测试代码，提高测试效率的经验。关键点是测试代码要仔细设计，测试需要围绕Interface进行；使用共享的Mock，而不是随意构建Mock对象；类的设计也要易于测试。

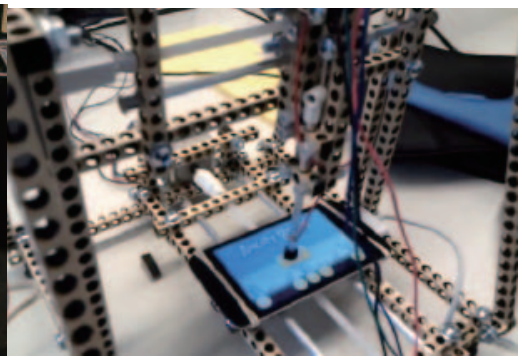
- Practical Machine Learning in Python：来自Hunch的工程师Matt Spitz以他的开源项目SluggerML为例，分享了用Python构建机器学习应用的经验，讨论了模型设计、数据处理、特征选择、预测和验证等问题。他使用的工具主要是NTLK和scikit-learn。

- Graph Processing in Python：以Python-Dev数据为例，Van Lindberg介绍了如何用networkx进行图分析。比较有意思的是Van Lindberg的职业并非工程师，而是专利律师，也是PSF的律师。

- pandas：Powerful data analysis tools for Python。随着Big Data的兴起，Python也越来越多地应用于数据分析领域。pandas是一个高性能数据处理和分析工具，其作者Wes McKinney介绍了pandas的特性和使用。

- Stepping through CPython：来自Python Core Dev的Larry Hastings介绍了CPython 3.2 runtime的重要内部实现细节，包括控制流实现、marshal、堆栈机、peephole优化、GC、Weakref等。

- Dismantling Jinja：Jinja2模板引擎作者Armin

使用Python编程的跳舞机器人和会打愤怒小鸟的机器人

Ronacher介绍了Jinja2是如何将模板DSL编译成为Python bytecode。

■ Building a Robot that Can Play Angry Birds on a Smartphone：Selenium的作者Jason Huggins介绍他怎么用Python、OpenCV和一些很便宜的材料搭一个会打愤怒的小鸟的机器人。

■ The Pyed Piper：pyp是一个用Python开发的文本处理工具，可用于日志处理、文本统计。该项目的作者Toby Rosen介绍了pyp的设计理念和使用方法。

■ Hand Coded Applications with SQLAlchemy：SQLAlchemy项目的作者Michael Bayer解释了这个项目的Toolkit开发哲学，并用具体的例子讲解了应当如何运用SQLAlchemy。

■ How the PyPy JIT works：Benjamin Peterson 介绍了 PyPy JIT 的工作原理和 JIT 观测工具 jitviewer。

■ Python Metaprogramming for Mad Scientists and Evil Geniuses：Walker Hale介绍了如何运用monkey patching和sitecustomize.py来改造现有的Python function和class。

■ Transifex：Dimitris Glezos介绍了他的创业项目Transifex，一个开源的基于Django的本地化系统。它可以帮助你管理软件项目的本地化工作。我还有幸担任了这个讲座的主持。

本次PyCon还有一些讲座很有意思，主要涉及高级OOP编程、分布式系统、Web开发和软件测试等领域。但限于篇幅，就不在此列出了。PyCon将在PyVideo.org上发布本次大会的部分教程和所有讲座的视频，地址是 http://pyvideo.org/category/17/pycon-us-2012 。

PyCon每天都有1~2次、每次时长1小时左右的启发性讲座。参会者可自由报名参加，用5分钟的时间来宣传自己的项目，分享开发经验，甚至调侃一下。虽然每个演讲很短，但内容丰富精彩，质量不逊于专题讲座。

大会的最后一部分是为期4天的Sprints阶段。150多名Python开发者在30多个开源项目中选择自己感兴趣的主题自愿加入，闭门开发，切磋技艺，不亦乐乎！

除了听讲座，和来自各地的开发者交流也是件非常有意思的事情。大会期间，我结识了几位分别来自上海、纽约和温哥华的华人互联网工程师，他们都很有经验，此外还认识了几位美国工程师。他们都在用Python从事有趣的工作，例如用Python和FPGA做高频交易、用Python分析药品和蛋白质数据、做电脑动画等。

这次规模空前的PyCon不仅展示了Python社区的最新发展，也体现了开源运动的力量和魅力。我相信，每一位Python开发者除了可以从PyCon中学习知识之外，还可以深切地感受到那种对计算机编程艺术的热爱和追求，而这正是推动软件技术不断发展的核心动力之一。



骆逸

毕肯互动（北京）科技有限公司创始人，从事搜索引擎和推荐系统的开发。在创办毕肯互动之前，曾任WPP ITOP 24/7公司CTO、空中网悟空搜索副总裁和美国Yahoo资深工程师。毕业于清华大学和美国亚利桑那大学。微博ID：@骆逸

大数据里的小利器

Strata 2012大会见闻

文 / 车品觉

如今，90%的数据分析师都在谈大数据，在大数据的背景之下，数据分析师收集数据的边界在哪里？又该如何运用数据呢？作者在美国参加Strata 2012大会期间，跟很多数据中人交流，其中前LinkedIn首席科学家DJ Patil给他的印象最深。

对话：先有需求，后有数据

车品觉：我有一个问题一直很困惑，现在的企业获取数据很容易，并且数据的增长速度非常之快，那么对于公司来说，到底要收集什么数据呢？收集多少数据？收集数据的边界在哪里？

Patil：过去收集数据很难，而现在获取数据资源比较容易。如果收集数据的出发点，不是为了解决问题，那么收集数据的量也太大了。

车品觉：可是许多公司认为，现在收集数据不难，成本也不高，为什么不先收集数据再说呢？等以后需要数据来解决问题时再拿出来用也可以啊。

Patil：千万别这么想，用这样的理念来设计数据产品肯定会失败的。数据是没有边界的，我为此也痛苦了好一段日子。比如收集一个人的生日，可以精确到几分几秒，但怎么用却不知道，那么这个数据就没有什么用。

车品觉：实际上，数据也是有生命周期的，比如从中国身份证号码是可以推断出性别的，但过几年如果这个规则变了，那么这个数据的基础就发生了改变，导致我们基于数据所做的假设和决策依据也就失去了意义（Data Broken）。并且，保存数据及其收集时的背景也不是一件容易的事情。所以说，在收集数据的同时，我们必须知道数据是用来做什么的，今天都想不出来，日后就更不容易想出来了。

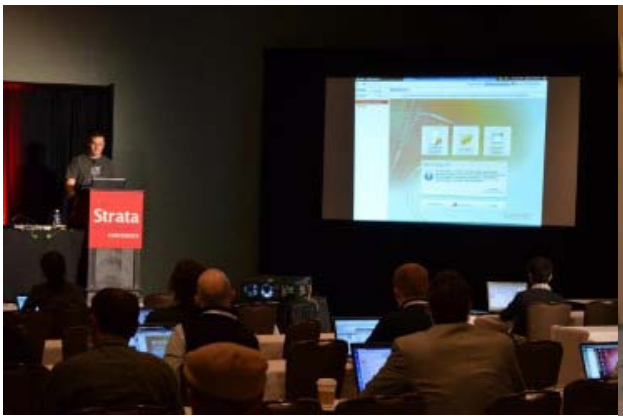
打个比方，今天很多电商老板会问重复购买率是多少，于是我们收集数据来计算重复购买率，却很少想到需要重复购买率来做什么决定。“刻舟求剑”的故事告诉我们的是世事在变，我们不能只是机械地套用方法或指标。就像重复购买率有不同的定义，而做不同的决策需要不同定义的重复购买率。如果A公司想收购B公司，那么关注的重复购买率可能细致到3个月内，购买一次的用户比例是多少，购买2~3次的用户比例是多少，3~4次的用户比例又是多少。如果A公司只是衡量自己的运营，可能它更关注的是日、周级别的重复购买率的变化趋势，或者当月新增客户有多少人是重复购买的，从而可以衡量每个月新增客户的最终忠诚度和质量。

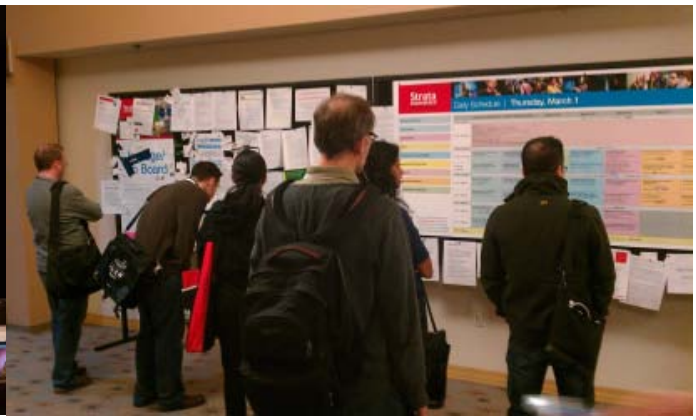
数据应用因小而美

我在做数据应用的这段时间，曾经特别为收集什么样的数据而烦恼。当时我想做一个特别大的数据应用出来，适合大多数人使用，可后来发现这在起步阶段几乎是不可能的。一是可以解决大部分人需求的数据应用根本就不存在，二是支付宝的数据非常丰富，需要考虑的因素很多，因素之间的联系又很复杂。

所以我总结，当做数据应用时，数据就等于原材料，当原材料一直处于变化的情况下，做出来的应用也有问题。体会数据和应用的关系之后，我决定从小角度切入，先做出小应用来。

这里说的“小”指的是应用的目标很具体。打个比方来说，对于一款数据应用，如果我的目的是





Strata 2012涵盖了从技术到商业等多个主题

分辨两种决策谁更好，差异在哪里，是很具体的问题。但如果我的目标是想知道如何让公司赢利，就是一个空泛的目标。

还请注意，“小”不是指数据量。许多人在没有获取足够数据，并且缺乏对数据理解的情况下做判断，其实是在享受自己的无知。

经过一番周折，我选择了按照小角度切入的想法设计数据应用，小角度切入设计应用可以做到具体和快速，而且可以避免因原材料的变化而导致的问题。

这次到美国来也有所感触，现在许多美国数据分析师都在谈空军打仗时采用的分析模型OODA（观察-调节-决策-行动），由于空军打仗强调快速决定，所以这个模型也特别适合今天互联网的需求。这个模型的核心思想就是快速出击，而对于今天的互联网公司来说，发展速度太快了，而数据分析师也必须在快速的发展环境中，迅速找出解决方案。

这个模式完全体现了互联网的快速找错、迅速调节的需要。快速实现原型，对于从来没有用过数据解决问题的互联网公司来说，从小角度切入才更加实际。大数据的背景之下，数据不仅量多，而且种类多。起步阶段，如果不从小角度切入，很难做出实用的产品和看得见的成绩。

把数据放进“框架”之中

这说到了另外一个话题，在大数据的背景下，必须考虑数据之间的关联性。一个单独的数据是没有意义的，实际中，选择处在两个极端的数据往往更容易找出它们之间的联系，把它们放在一个框架中看才能发现问题。

举例来说，我曾经研究美国有哪些网站值得我们学习。依靠数据寻找美国的互联网应用黑马，便是从问题的角度出发。在各种数据中，我选择了“访问量”和“停留时间” 这两个往往此消彼长的数据作为帮助我决策的框架。利用这个框架，我在2010年就发现了Pinterest，远早于国内对它进行模仿的应用。

因此，关于如何在大数据的背景下，利用数据做决策，我总结了四步走的方法：第一，从解决问题的角度出发收集数据；第二，把收集的数据整理好，放入一个框架内，并利用这个框架帮助决策者做出决定；第三，评估决定与行动的效果，这将告诉我们框架是否合理；第四，如果有新的数据出现，我们将考察能否利用它对前面三步做出改进，以及我们今天是否还需要收集更多种类的数据。



车品觉

支付宝首席商业智能官。中国香港人，曾在美英澳洲等地接受西方教育，清华及INSEAD双硕士学位，加入支付宝前曾在敦煌网担任首席产品官。

强大的密码胜于利剑
RSA 2012参会心得

文 / 潘宣辰

作者第一次参加RSA大会，从中感受到了强烈和广泛的产业气息。希望本文能够从一个RSA新人的视角向读者展示一些有价值的信息。

有趣的主题Logo

在会场首先感受到的除了强烈的展会气氛外，最醒目的就是本届RSA大会的Logo了，官方对这个图标的描述是“The great cipher mightier than the sword（强大的密码胜于利剑）”。在一个以商业交流和展示为主的大会上，组织者打出这样的口号和Logo，让我顿时感觉少了一些商务味，多了几分亲切。

健全的美国信息产业

对我来说，印象最深的是整个美国信息安全产业的健全。在RSA展会上我看到的不仅是一个完善和规范化的市场，同时也看到美国同行做事的专业。不同厂商都有着自己对安全的独特见解和解决方案，大家从不同角度选择切入点，创造性地提供自己的服务，找到自己的盈利模式。有大如Symantec、McAfee这样的安全巨擘，也有许多只做某一细分领域的小安全厂商。有专注在网络安全设备领域的，也有专注在企业级安全解决方案的，还有个3人团队独立开发一个小安全Box。应该说，本次RSA大会传递的最强烈的信息就是全球信息安全市场是庞大的，虽然不同地域有不同的安全厂商，不同的安全解决方案，但安全的精神本质和技术方向是一致的，国外同行在保有自己独特的安全观点和创造性的安全解决方案同时，在同一个大舞台上展开良性竞争，这才有了RSA展会的盛世景象。反观国内，有特点或者有一些独特安全见解的厂商则相对较少，或者说依然并没有获得很好的发展机会。对于我们国内安全厂商来说，不妨切换到全球市场，也许能够发现更大的机会。也希望随着中国信息行业的不断发展和健全，能够给所有人提供一个更好的表达自己安全观点，实现安全理念价值的产业环境。

活跃的细分领域市场

在这次安全展会上，大部分时间我都在与不同安全厂商和同行进行交流。在与诸多安全厂商的交流中，我印象颇为深刻的是有5-7家做恶意代码Sandbox的公司，其中既有我从未听闻的公司，也有之前耳熟能详的Sandbox专业户。应该说，在PC恶意代码Sandbox领域看到这样的商业景象是我完全未料到的。可能在我们的设想中，PC恶意代码发展至今，Sandbox的市场和需求应该不会太旺盛了，而我在展会上看到的是，这几家厂商都有自己独具特色的产品理念和形态。有的Sandbox重点在性能，有的重点在行为分析有效性，还有的重点在0day发现和识别上。我看到的是一个非常健全的Sandbox小产业形态。在和他们的交流中，我听到的是一个此前从未想过的一个安全市场细分领域。他们的观点是，虽然现在的安全公司已经有了多年的发展和积累，但与恶意代码的对抗形势和方式始终在不断变化，恶意代码的威胁和局势与10年之前相比并没有减弱。他们从Sandbox产品出发，利用自己的技术创造力来解决某一方面的问题就能获得用户和市场的认可。而同时，在他们的视野中，这个市场很庞大，除了桌面反病毒产品安全公司、网络设备安全厂商，美国非常活跃的学术界对Sandbox的需求也很旺盛，从这样多个层面来看，他们看到的

是一个非常庞大的Sandbox产品需求市场。

反观国内，我认为不论从技术还是产品研发层面，与国外这些安全厂商的差距并不大。但整个安全行业的风气，仅与美国Sandbox这一个细分领域相比，都有着不小的差距。我认为，这是技术研发上的短视造成的，过去10年，中国安全行业发展迅速，但很多环节并不是良性的。从某种程度上来看，我认为核心竞争力反而在下降，这一点需要所有的传统安全厂商思考和努力。

手机安全的趋势

我的工作主要是从事手机恶意代码研究，因此在本次展会上对手机安全相关厂商非常关注。今年RSA创新论坛的头名就是以手机App安全服务为主要产品形态的一家创业公司。和以往相比，手机安全的戏份确实有所增加，但并没有达到我预期中的火爆。虽然不少安全厂商展示了自己的手机安全解决方案，但大多定位于企业级。此外他们的技术和产品新意不多。RSA缺少出名的手机安全厂商参与，这点颇为可惜。

从本次展会看，我认为许多安全厂商在手机安全领域的投入并不是特别积极，尤其是在手机恶意代码自动化分析、手机恶意代码检测引擎等相关技术环节几乎没有看到太多给我留下印象的内容。反倒是我们展示的内容引起了不少国外安全厂商和参会者的注意。

当然，我想这和手机安全在美国安全市场的形势和具体的地域局势有关。在和一些国外同行的交流中不难发现，手机恶意代码在国外并不严重，与国内相比，形式也有着巨大差异。我可以比较“自豪”地说，国内的手机恶意代码环境还真是“全球领先”。国外企业制度、法律法规完善，而且有着规范的移动互联网市场，带来的也是更加规范和干净的大环境。反观国内，手机应用发布渠道多样，手机软件的篡改、盗版、恶意代码捆绑屡见不鲜，因此使得中国成为手机恶意代码的重灾区。国外的手机安全市场需求目前主要集中在企业，强调移动终端MDM、企业移动终端介入控制、数据泄露保护和移动终端接入的策略管理等环节。而国内并没有形成良好的商业模式和产品模式，安全厂商主要依靠产品研发抓住终端客户。也许，我们也可以尝试在企业级市场加大投入，加强企业对移动终端的安全、数据的管理功能的研发。



RSA 2012会场交流气氛热烈

中国安全团队的表现

本次展会中，中国安全团队的参展数量也非常可观。比如我熟悉的东软、启明星辰，以及第一次参加RSA大会的金山。应该说，得益于中关村的组织，我们才能够在RSA的展会上占据中心厅位，并全程参与了展台布局等工作。这次参展，安天展示了自己在反病毒引擎和手机恶意代码分析方面的相关技术和产品，并带来了新版的安天病毒通缉令桌游。我们计划在2013年继续加大参展的力度，谋求可能的海外合作与商务机会。

总结

诚然，不论在展台布置、展示内容还是形式上，我们与国外厂商都还有着一定差距，但我们也能从中学习和借鉴，通过参加RSA展会，国内安全厂商也能够取得很好的海外推广和谋求商业合作的机会。希望2013年的RSA能出现更多中国安全厂商的面孔，让我们也能做一回RSA的主角。



潘宣辰

安天实验室武汉研发中心总经理，毕业于武汉大学信息安全专业。目前主要从事手机恶意代码检测引擎和自动化分析系统的相关研发工作。

跨平台游戏引擎 Unity3D初体验

记者 / 卢鹄翔

Unity是一款用于创建三维视频游戏、三维动画等互动内容的综合型创作工具，具有出色的跨平台兼容性。2012年4月，Unity开发者大会将围绕移动平台游戏开发的方方面面做全景性展示，为国内游戏开发者带来第一手经验分享。

什么是游戏引擎

“游戏引擎”这个术语最早出现在20世纪90年代，当时id Software的第一人称射击游戏《毁灭战士》的软件架构已经相当清楚地划分为核心软件组件、游戏资产、游戏规则等几部分。其中的核心软件组件包括渲染器、物理引擎、碰撞检测系统、音效、脚本引擎等部分，这就是我们今天所说的游戏引擎。这样划分的重要价值在于，因为开发者在得到引擎部分授权后，只要重新制作美工、调整关卡布局、设置游戏规则，就可以打造一款全新产品，而游戏工作室对外授权引擎，也成为他们第二个收入来源。如今游戏开发者大多选择购买一个游戏引擎授权，复用其中大部分关键软件制作自己的游戏。虽然仍有不少工作要做，但比起从头开发，任务已经轻松了很多。

如今，游戏已经是百亿美元的产业，规模不逊于好莱坞，利用游戏引擎，开发者几乎能实现任何自己想要实现的游戏。

跨平台的游戏引擎

有些人可能以为游戏引擎能变成一个通用软件，运行于任何平台，可以制作任何想象得到的游戏内容。然而，这个设想并不容易实现。大部分游戏引擎针需要对特定游戏及特定硬件平台精心制作和微调。就算是一些最通用的游戏引擎，其实也只适合制作某类型游戏，例如第一人称射击或赛车游戏。

这是因为设计高效软件总要面临取舍。例如，一个为室内环境设计的渲染引擎，一般不能很好地渲染室外场景。不过随着电脑硬件速度的提高和专用显卡的应用，加上软件不断优化，不同类型的游戏引擎差异慢慢在缩小。但通用性和效能仍然需要取舍，经常可以通过微调引擎去制作更精美的游戏。

Unity是一款完整的游戏开发引擎和工具，其核心是基于C/C++语言。Unity的开发环境是Microsoft Windows和OS X，所用开发工具是微软的Visual Studio（在Windows系统上开发）和Xcode（在苹果机上开发）。它的渲染系统是基于Direct3D、OpenGL、OpenGL ES，以及单机系统的API。Unity内的游戏逻辑脚本基于一个开源的.NET架构Mono。Unity也集成了许多当今优秀的插件，如PhysX、Lightmapping、Occlusion Culling、Pathfinding等。

Unity公司最大的市场依然在美国，但亚洲地区2011年的增长达到了270%。Unity增长的同时带来了用户的基础，最初这个游戏引擎多半是被独立游戏开发商、小型团队采用的，而现在变得越来越受到大型公司的欢迎。这个局面的出现，让Unity公司改变策略，将这款游戏持续发展，变成一个通用引擎，变成一个如同Epic的Unreal引擎一样的重量级选手。

和其它游戏开发引擎及其工具相比，Unity在跨平台开发方面有着一定优势。即把同一款游戏发布到不同平台，在同一个编辑器中就可以完成。而且Unity的编辑器上手容易，加之Unity提供的大量的可用资源，使得游戏开发周期缩短了许多。

游戏开发团队的组成

游戏研发是一个包罗万象的系统工程，在各类软件工程中，游戏开发是对开发者要求技能最高、最全面的一类。所以对于研发工程师来讲，除了对每个功能的基本了解之外，还要有总体架构的设想。这样才能打造一个完善的系统。

游戏工作室通常由5个基本专业领域的人员构成，包括工程师、美工、游戏设计师、制作人及其他管理支持人员，下面简单谈谈前3类。

■ 工程师负责设计并实现软件，使游戏及工具得以运行。有时候，工程师分为两类：运行时工程师和工具工程师。运行时工程师制作引擎和游戏本身；工具工程师制作离线工具，供整个团队使用，以提高团队的工作效率。运行时、工具两方面的工程师都各有专长。

■ 游戏界经常提到：“内容为王”，美工就肩负制作游戏中所有视听内容的重任，而这些内容的品质能够决定游戏成败。概念美术设计师以素描或绘画，让团队了解游戏预设的最终面貌。概念美术设计师的工作始于游戏开发的概念阶段，一般会在项目的整个生命周期里继续作美术指导。三维建模师为游戏世界的所有事物制作三维几何模型。纹理设计师制作称为纹理的二维影像。这些纹理用来贴附于三维模型之上，以增加模型的细节及真实感。灯光师布置游戏世界的静态和动态光源，并通过颜色、亮度、光源方向等设定，加强每个场景的美感及情感。此外还有动画、音效等相关的负责人员。

■ 游戏设计师负责设计玩家体验的互动部分，在宏观层面上设定故事主线、整体的章节或关卡顺序、玩家的高层次目标。有些游戏团队会聘请一位或多位作家与游戏设计师合作编制故事主线，以至编写每句对话。

使用Unity引擎的经典游戏

CMune公司的第一人称射击游戏《UberStrike》仍然是排在前面的一款用Unity开发的游戏。这款游戏部署在Facebook平台上。MadFinger公司开发的《ShadowGun》充分发挥了Unity在移动平台方面的优势。韩国公司NDoors开发的《The Romance of The Three Kingdoms》也是一款很有竞争力的游戏。中国开发者开发的已经上线的游戏，有四川九众互动开发的《将魂》和上海老友记网络开发的《猪仙》等。

移动平台的游戏引擎

相对于桌面平台，在移动平台上开发游戏会受到较多限制，主要在于存储空间和CPU、GPU的处理能力。其次是系统的规范化。比如Android系统，生产厂家很多（特别在亚洲，中国地区），而



Facebook平台下使用Unity3D引擎开发的《UberStrike》

且都对原有的系统进行了一些修改，或者增删一些功能。这样游戏引擎就必须对所有使用者可能用到的硬件做一系列测试。对于Unity来说，引擎开发者需要找到这些硬件来进行测试，才能保障游戏开发者顺利地进行开发。

Unity在移动平台上实现了一套完整的游戏开发引擎和工具，其中包括开发者所需的渲染功能（包括灯光、阴影等）、动画功能、物理特效、粒子特效、地形系统、寻路系统、音效、脚本编辑功能、基本的网络接口，以及方便灵活、简单易学的编辑器。使用者可以很方便地对引擎及其编辑器进行扩展。使用的编程语言可以是C/C++、Java（写插件）、C#，或者用JavaScript、Boo来写脚本程序。

借用Unity创始人的话，Unity就是要打造大众化的引擎，让每个人都能很容易地使用它来开发游戏。虽然Unity基于移动平台，但并不意味着只希望提供“移动领域的开发工具”。Unity新的推广举措是，到4月8日前，使用Unity游戏引擎附加许可证开发iOS和Android游戏将免费（这一推广措施，可以为开发者节省800美元的授权费用）。

还值得一提的是，3D引擎的应用范围并不仅限于制作游戏，它还常被应用在教育、建筑、商业展示等领域，为用户带来丰富立体的演示效果。Web平台上的《保时捷911选购工具》，以及iOS上的《奥迪风云天气预报》等程序，都是使用Unity 3D引擎开发的商业应用。

2012年，Unity宣布将在亚洲举办一系列开发者大会。亚洲的游戏开发者势力越来越强大，通过这一系列的开发者大会，开发者们可以与Unity进行更快速和有效地沟通。4月中旬，Unity开发者大会将首次登陆中国，为广大的3D游戏开发者带来为期两天的第一手的游戏开发经验分享。

创 业



对于互联网从业者而言，创业意味着你将经历一次自我修行，全力地追逐梦想，并有机会创造财富甚至改变世界。同时，也意味着你和你的团队将独自应对所有始料未及的挑战。本期封面报道，我们将从创业方向、创业伙伴、技术选型、融资等几个角度，探讨互联网创业的方方面面，以求让你的创业之路，不再孤单。

正在改变世界的硅谷创业趋势

文 / 骆轶航

基于云计算的办公革命

如果你对云计算应用的理解还停留在同步和共享个人私密文档的层面上，那么你可能会错过很多正在发生的激动人心的变化。

基于云环境的办公软件提供商Box是一家创建于2007年的公司，位于帕罗阿图。其创始人兼CEO Aaron Levis预言说：未来18~36个月，人们的工作方式将发生根本的革命性变化。

这个变化首先是从办公软件开始的。Box的诞生至少证明了一点：那就是在这个时代创办一家办公软件公司并不是一件很糟糕的事。未来的办公可能不属于微软的Office，可能不属于甲骨文和SAP的CRM系统，但它或许属于Box。

看看这家创业公司做了什么：它是一套完整的在线办公软件，具有强大的协作性，这表示它也是一套协作工具，更重要的是，它是一套大多数人都可以访问的公共云存储器。这意味着过去你至少需要一台服务器、一套Office办公工具和一个微软SharePoint协同工具才能勉强构建起一套协作式的工作环境，但现在Box把它们一次性地解决了。更重要的是，Box的架构具有高度的开放性，可以支持包括同样基于云计算的Google在线文档、办公和客户关系管理工具Salesforce，以及任何一款企业社交工具Yammer和Moxie的接入（这些都是硅谷近2~3年来涌现出的创业公司），也就是说，它是那些安装在你电脑和服务器上的应用程序的终结者。目前Box宣称已经有70%的全球500强企业是它的客户。

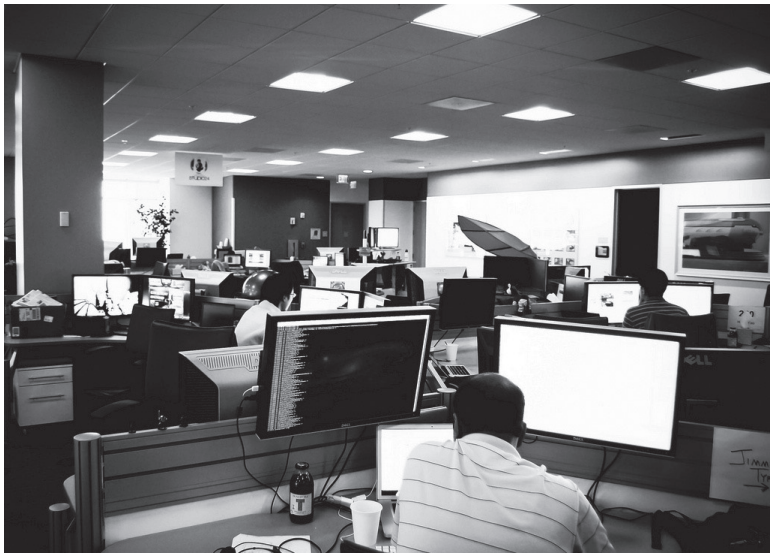
这个即将在未来18~36个月内引爆“办公革命”的趋势还体现在更多的方面：由于企业办公系统和数据中心、企业社交网络和云端的客户关系工具等被无缝地整合在一起，它释放了人们的办公空间和办公方式。让人们更倾向于“自由人自由联合”的工作状态——越来越多位于硅谷的大公司和商业楼宇代理商留出了“灵活流动”的办公室和空间，留给那些需要在不同场合与不同人一起工作的人们——今天与客户，明天与供应商或合作伙伴，后天与其他部门的同事。而另一些人则选择了灵活办公的方式，而不用每天赶到同一间办公室去。它同时催生了基于地理位置的寻找灵活办公空间的服务的诞生——LiquidSpace就是其中典型的代表。

这场办公革命还催生了另一个创业趋势的兴起——基于移动设备的视频协同会议系统。某种程度上它又是思科网真等大型笨重的演播室视频会议系统的终结者。办公空间的灵活和移动设备的普及，让能提供相对复杂协作、同时又维持清晰视频与交互质量的移动应用型视频会议服务变得越来越重要，而这一领域近两年也陆续诞生了诸如GoToMeeting、FuzeMeeting、MightyMeeting和Twivilo等一系列初创型企业。

“群体智慧”的众包服务

对一家公司或机构来说，并非所有的工作事项都得由内部员工和指定的供应商完成；对于个人而言，也并非全部的需求都得通过某一个固定有组织的交易平台才能被满足。每个人都可能成为某项“特殊使命”的执行者，每个人都可以成为一个交易平台，每个人都可能成为别人的导师和助手，这是群体智慧的力量，也是硅谷工程师们创业的热门方向——如何利用群体智慧“众包”人们的需求，创造商业的价值。

其中最著名的例子是Zarrly。这是一家2011年刚



硅谷AOL大楼已经成为创业者的乐园，CloudOn就是其中之一

成立的公司，它为用户提供了一个基于地理位置的、实时的、买方驱动的移动交易市场——几乎每个人心中都对每样东西有一个价格定位，而Zarrly就是促成这种交易。它完全由买方驱动——你可以发出邀约，出50美元寻找一个附近的人帮你到苹果专卖店里排队买一个新iPad，也可以开出30美元的价格邀请附近的一位高材生帮你解答一道复杂的微积分题目，或者出更高的价格请精通厨艺的附近的邻居为你烹饪一道地中海菜。这些交易在以往的市场中可能并不存在，但Zarrly创造了它们。

另一个类似的公司是Gigwalk。现在，很多大公司都通过Gigwalk的平台发布琐碎而繁杂的任务，而安装了Gigwalk手机客户端的用户，只需要完成这些相应的任务，符合买方的要求就可以得到相应的报酬。在创办初期，这家公司的任务大多是那些简单的“跑腿”型任务，比如帮Yelp收集餐馆照片，帮导航工具TomTom收集线下路况以及为微软Bing搜索引擎提供景点采集等。而随着时间的推移，一些智力含量更高的任务（gigs）开始出现在这个平台上——比如编程和代码外包等。它的数万名用户在过去一年里已在这个平台上赚取了上万美元的收入。

某种程度上，知名的社交问答网站Quora也是这种群体智慧和众包路线的产物。值得注意的是，创业者们在众包的细分领域似乎仍大有可为。之前有Waze作为典型的案例——它让司机在驾驶的时候开启GPS，通过智能手机的近距离传感实现语音控制，用户可以通过语音互相报告交通情况和提供导航互助，最终绘制成用户自发生成的交通路况地图。而最新的众包智慧案例发生在教育领域——OpenStudy是一家新成立的教育社交公司，你可以把它看作是中学教育的Quora，用户可以在上面发布不同学科的复杂题目，然后有其他的用户负责解答并获得积分。与Quora不同的是，为了强调“群体智慧”的效果，它采用实时的方式，大部分问题将在被抛出之后立即得到解答。

数据与物理设备相连接的未来医疗

也许移动健康与医疗会让乏味了很久的硬件设备领域重新焕发活力。因为几乎每一个与健康相关的App都最终将与人们的身体机能发生关联——它需要传感装置去捕捉关于人们身体的各种数据：无论是心跳、睡眠时的脑电波，还是运动中的脉搏周期，也无论这个传感装置是一部手机、一个微型测量仪和一条腕带。

2011年夏天刚创建的移动睡眠管理应用LARK就是一个典型的产品——它是一款iOS应用，用来监控人们的睡眠质量。除了作为记录与分析数据枢纽的应用程序之外，它还需要一个传感器和一条系在手腕上的腕带——前者通过蓝牙装置将用户睡眠时的生理体征传送到手机里的应用程序，后者是直接与用户身体发生接触的物理传感器。LARK的独立设备——传感器和腕带，被苹果的零售部门挑选为重点推荐的周边产品，摆放在加州任何一家苹果专卖店的显著位置。

类似的产品还包括著名的Fitbit。Fitbit成立于2008年，它的产品是一款应用和一系列外接装置的组合。第一款Fitbit的外接装置是Fitbit Tracker，它是可以随身携带的芯片夹测量计，夹在身上的任一部位就可以将当天包括移动距离、移动脚步数量、卡路里的消耗量以及身体的活跃时间等各项数据记录下来。第二款产品是Fitbit Ultra，主要添加了睡眠质量监控功能。2012年它又发布了一款新的外接产品——Fitbit Aria的无线体重秤，能支持多达8名用户的数据储存和识别功能，适合家庭

和团队使用。

也有人在尝试将移动终端本身当作传感器。新近创办的老年人健康状况追踪应用Elderaware就是一例，当这款程序开启时，它能利用重力感应装置，准确辨别用户的地理位置和身体状况。老年用户不慎摔倒或晕厥时，它能根据手机本身异常的反应而发送警报到呼叫中心，从而发起必要的紧急救助服务。然而由于用户行为方式本身的复杂程度，Elderaware创始人Rick Sheridan本身也考虑未来支持独立的设备，以更准确地监控和响应老年用户的生理和健康突发情况。考虑到保健和医疗将是一个拥有巨大需求的市场，这一领域的移动应用和独立的硬件设备需求将迎来井喷。

而另一个对独立硬件设备需求庞大的领域是移动支付：无论是PayPal最新的数字钱包和新兴支付平台Square对硬件的需求，还是更新兴的移动支付工具Level Up对传感装置的特定设计，独立硬件的作用正在凸显。这意味着传统的硬件工程师和测试工程师将更有机会在一家创业公司中扮演核心的角色，甚至创办一家新的公司，这并非软件工程师的专利。

下一代知识管理与搜索引擎

信息结构与算法仍是硅谷顶级公司和创业团队最推崇的事，它决定了一家创业公司能站得有多高。那些可能成为下一个Google和Facebook的团队通常与这一领域密切相关。幸运的是，除了SoLoMo和电子商务那些炙手可热的项目之外，硅谷的一些团队从未放弃对这一领域的追求。

2008年创办的云笔记同步工具Evernote因为逐渐成为人们托管记忆数据的外部“大脑”而倍受青睐，成为人们管理日渐庞杂与琐碎知识信息的工具。而现在有更多工具将实现人们对知识更深入挖掘和管理的需求。例如一年半之前成立的知识分享与管理工具服务商Pearltrees——它本质上是一个开放和协作的网络兴趣图谱，但同时也是一个知识管理工具。它将你储藏的不同信息源和知识源装在不同的“气泡”里，由你自己创造“气泡”之间的逻辑与架构，就像树的枝蔓那样延展开来，实现分类管理知识和信息的需求。而其他的好友与协作者也可以根据你的知识分类枝蔓搜寻和“摘取”他们需要的知识气泡，放在自己的气泡图谱当中。它的乐趣在于对知识管理的分类与发现过程，以及在某一个知识门类之下，与他人针对信息的协作与分享方式。

毫无疑问，Pearltrees的出现依赖更为复杂的信息架构以及算法——人们在不断动态变化的信息分类与组合，以及与他们共享的信息图谱当中搜寻他们需要的信息。它本质上是一个信息搜索的算法引擎。事实上，这类强调算法、搜索精准与匹配度的公司在硅谷还有很多，有些甚至不为人知——比如为Facebook的社交化News Feed精确匹配推送提供算法和背后引擎的，就是一家叫做PageLever的小型创业公司。

似乎“终结Google”的下一代搜索引擎革命也可以预期。当大多数的互联网体验开始从PC转移到平板电脑和智能手机等移动设备上时，Google当年创造的基于PC网页的搜索技术就显得有些力不从心了。而最近出现了一家新的搜索引擎公司Everything.me似乎正在将搜索的未来引向另一个有趣但不可知的方向上：它专门为移动设备和平台设计，搜索的结果呈更可视化的图标状排列，颠覆了传统搜索的链接排列风格。更重要的是，它将为每个人提供定制化的搜索结果。尤其是当人们的日常搜索通常更倾向于围绕人、地点和具体的事物，在移动设备上进行时，Everything.me好像真的让搜索变成了另外一回事。

当然Everything.me不会是下一代搜索引擎唯一的样板。硅谷天使投资教父Paul Graham近日在《雄心万丈的创业思路》的博客中罗列了未来将发生颠覆性创新的几个领域，排在第一位的就是搜索引擎。他认为：“在10多年的时间里，换个搜索引擎（即使不用Google）的事在今天第一次变得可以想象了。”而在他做出这个论断之前，人们已经看到了诸多跃跃欲试的前仆后继者。



骆轶航

《第一财经周刊》主笔，驻旧金山硅谷代表。关注硅谷最热门的创业趋势和成长型公司，侧重移动应用开发、云计算、社交游戏等领域的发现与报道。

如何挑选联合创始人

文 / Elad Gil　译 / 陈向伟

初创公司失败的最主要原因不外乎资金枯竭和创始人之间起冲突，本文给出了一些挑选联合创始人的建议。

在创办Mixer Labs时，我对联合创始人的选择是非常幸运的。当你决定创办一家公司时，挑选联合创始人是一项最为重要的决定。你的联合创始人如同你的配偶，每天的大部分时间你都会和他们在一起，在所有重大问题上你还需要和他们达成共识（打造什么产品、如何发布、是否出售公司……）。

此外，联合创始人也将是建立公司文化和职业道德的基础。连同你在内的一两个人的团队在刚开始将决定初创公司所作的一切工作，还有同样重要的，就是如何去做这些工作。

以下就是我在寻找一名伟大联合创始人中遵循的最高标准。

相互沟通的能力

你与你的联合创始人可以进行诚实和坦率的对话吗？当他们做出了错误的决策或为了错误的理由开展工作时，你会告诉他们吗？你能不能给出诚实和批评性的意见，你能否接受他们诚实和批评性的意见？如果你出于个人利益做了一些决定，你在为此说明理由时能和他们坦诚相待吗？

初创公司的压力很大。尽管压力存在，但你需要和联合创始人进行坦诚的对话。如果你们不能相互交流，不能从错误中吸取教训，那么你的初创公司也不会进步。

上面所说的一切都是基于相互尊重和信任，并要认识到你们是在共同战斗。如同任何关系一样，联合创始人关系会经历很多的起起伏伏。一路颠簸走来，其中的关键就是相互沟通。

厘清关键问题

做“朋友”和与他们一起创业完全是两码事。

你们的目标是什么？为什么要创业？人们会因为很多原因开办公司。有些人是想赚大钱，有些人是为了改变世界，有些人是为了成为网络名人。你要确保联合创始人和你的目标相同。否则的话，你可能因为500万美元，就接受Google的召唤退出，而你的联合创始人却希望干出一番大事业，创业合作随即终止。

你们想要的公司文化是什么？你们希望公司文化中拥有的3~5个关键点是什么？你认同他们的观点吗？你们希望聘请的是多面手还是专家？在招募新员工时，你们认为最关键的3~5个特质是什么？

你们对投资者的看法是否相同？投资者在公司中的角色是什么？我的一位首席执行官朋友遭到解雇，因为在一项很重要的战略决策中，他的联合创始人与风投站到了一起。他的联合创始人为什么这么做？因为他觉得一旦风投投资了公司，那么就是他们来做战略决策了。这是一种天真的看法，但一些人就是这么想的。你必须搞清楚联合创始人的想法。

基于角色的协定

许多大公司早期都是由一名创始人主管或担任首席执行官（例如史蒂夫·乔布斯、比尔·盖茨、拉里·佩奇、马克·扎克伯格），另一个人担任技术主管或配角（史蒂夫·沃兹尼亚克、保罗·艾伦、谢尔盖·布林、其他多位Facebook创始人），在这种模式下这些公司都获得了明显



乔布斯和沃兹尼亚克

的突破。

如果你们两个都想成为首席执行官（作为妥协，其中一个一般会被授予“总裁”头衔），你们同意在做所有重大决策时，两人的地位是“平等”的，这样你们就要准备好发生冲突的可能性。在创业初期，决策的出炉显得非常容易，你和你的联合创始人也配合得很好，因此你们认为这种模式很管用。丑话说在前头，在谁发号施令和谁做出最终决定问题上没有明确的话，你们就在为未来的冲突埋下隐患。

即便你们在95%的事情上都达成共识，但只要有5%的异议就可以让公司瘫痪好几个月。你们可能都声称拥有否决权，这意味着无法达成共识，公司无法获得发展。这对初创公司就意味着死亡。对于初创公司来说，所有的核心前提和竞争优势就是具备快速行动的能力。

在创办公司之前，你就要明确一位决策制定者。不要把这么痛苦的沟通拖到公司创业早期。我的拙见是，假如有必要的话，在99%的情况下，首席执行官都应该是决策执行者，不管是商业问题还是产品问题。

你们必须预先商定解决冲突的办法。例如“我们的外聘顾问梅丽莎可以来做调解。”

创始人之间可能会出现相互竞争的局面。首席执行官通常被认为是受邀出席会议的人，他可以吸引到风投（这不算迷人，但大家就是这么想）。如果两个人都想当首席执行官，那个没有当上的人可能会因为没有获得足够的曝光率，而认为自己获得了不公正待遇。

这种情况会随着时间的推移暴露出来。也许你拥有最不起眼又非常好心的联合创始人，但六个月后，他们可能会开始感觉自己被遗忘了。如果发生这种情况，你要试图找到一种方法，既要在首席执行官角色上不妥协，又要增加他们的曝光率。

技能不重叠（除非都是写代码的）

通常一家初创公司只会有一件产品。两个人制定产品前景的结果通常是，要么让专业的团体来设计，要么在发展方向上无法达成共识。这并不意味着要找个对产品感觉不良的人。相反，它意味着你需要一个对产品具有强烈感觉的人来设定发展方向，并且要对他设定的目标明确认同。

相似的是，三位商界人士创办一家科技公司通常都是一场灾难。通常不会有足够的“商业业务”让三位商业联合创始人来做（除非相比科技公司来说，公司大量的工作是品牌建立/营运，比如眼镜折扣网站Warby Parker、化妆品网站Birch Box等，在这些公司就很容易开拓角色）。

唯一有用的重叠技能就是所有创始人都是开发者。即使是这样，也要明确每个人的分工（比如谁来当首席执行官）。

最好具备的一些条件

联合创始人能拥有以下特质那就最好了。

已经相互认识好多年。你们相互认识、尊重并信任，这将有助于进行更为开放的沟通。所以你明白获得的建议或进行的艰难对话都是善意的。

曾与他们直接共事。在生活和工作中人们的表现可能完全不同。你最好和他们在相关的业务领域或其他项目中合作过。一旦艰苦的工作接踵而至，又涉及钱的问题，人们的态度会发生很大的改变。如果你们曾经共事过，那你们就彼此了解对方的工作风格和节奏，所以工作效率也会更高。这样你们也就能更好地定义一起工作中的角色，并信任彼此的角色。

（本文作者Elad Gil，是Mixer Labs创始人&CEO，曾在Google负责无线团队孵化工作。在此之前，还曾任职于麦肯锡，目前担任Twitter战略总监。）

创业公司如何确定技术路线

文 / 郝培强

首先要说的是，我认为创业公司的技术路线选择并不是一个技术问题。如果是一个纯粹的技术问题，那么很可能会一直争论不休，最后的结果就像某个笑话里面说的，某公司一直到倒闭技术路线还没统一。因此，本文不会向你推荐任何一个具体的技术方案。

创业公司往往都面临着不同程度的资金和资源短缺，经不起反复修改技术路线带来的资源浪费，需要管理者或者技术带头人能够较早地确定一个适合的技术路线。因为技术路线决定了岗位如何设置、人员如何招聘、开发周期、开发模式等。而更多时候，人员招聘可能是最核心的问题。

那么创业公司该如何选择技术路线呢？我有如下一些看法。

给非技术出身创业者的建议

以前做技术咨询时，我们经常会遇到一些非技术出身的创业者，他们往往对好的技术人员非常渴求。但对如何找到好的技术人员，如何吸引技术人员加入，他们却束手无策。在我创业的这两年，这点得到了充分的印证，我可以轻松地劝喜欢的技术人员加入，但要找到好的设计师并说服他们加盟太难。各种人才都有自己的圈子、文化和行为方式，他人很难融入。

对于非技术出身的创业者，我的建议是，尽量使用开源系统、现成的解决方案。虽然技术人员往往对Discuz!、UcHome甚至Wordpress等现成的系统颇有微词，但对于没有技术背景的创业者，这些系统是最好的切入点。

我见过一些非常成功的团队，创始人完全不懂技术，但对产品、运营、营销非常在行，他们的网站全部采用开源系统，没有任何的改动，只是找了设计师对样式做了修改。网站的流量、用户满意度都非常不错。因为没有任何先行的开发，所以网站可以很快正式上线，而且大多数程序的开源系统，也都有较强的自定义能力，仔细琢磨下就可以玩出很多花样。

这样的团队往往没有技术人员，或者只有一两个很初级的技术人员，因此，在网站成功发展后，往往会遇到一些技术瓶颈：例如在流量猛增后，技术人员不知道如何化解；产品发展了，开源软件的基础扩展能力不能满足需求以后，需要二次开发或者需要重新开发时，现有技术人员的能力无法跟上。

这时就需要考虑引入一个好的技术带头人，建立自己的技术路线。通常也会有一番波折，但这也是种甜蜜的烦恼，因为不发展到一定程度，根本不需要考虑。发展到这样的程度，公司的现金储备和知名度都有了很大的提升，招人也就会容易一些。

给技术出身的创业者的建议

首先，我认为一切企业管理的出发点和归宿都是“人”。什么样的公司吸引什么样的人才，所找的人才又会影响公司企业文化的形成。技术路线的选择，一定要结合获取人才的难易和开发模式的组织情况来思考。

激进的技术路线

很多人喜欢激进的技术路线，比如用Erlang、Go等语言做项目，而公司全部员工几乎都是初学者。又或者有人在iOS第一代SDK出现时就投身

App Store。

有时，有些功能或产品只有激进的技术路线才能解决，较早使用先进技术可以获得技术优势和壁垒，在一般网站都使用SQL Like语句做全文搜索时，如果你率先使用Lucene/Sphinx等开源搜索引擎，那么就会在性能和效果上有非常明显的优势。技术发展非常快，但普及非常慢。在2001年开发出Lucene第一个版本后不久，国内的技术人员就开始写文章和学习笔记介绍它，但到了2008年，国内很多中型网站的全文搜索仍在使用SQL Like语句，很多技术人员明知这样耗费服务器资源很严重，甚至会拖垮服务器，但也不会改用Lucene。

有时，激进的技术路线可以帮助你在新兴市场占领先机。前两年苹果App Store刚刚出现，软件比较缺乏，所以只要产品质量尚可就可以获得很多用户，从而获得不菲的收入。现在开发者想做出被市场认可的产品非常难，而那些进入早的产品则可以靠早期积累下的用户稳定可靠地发展。

激进的技术路线的另一个好处是，可以作为人才过滤器。如果公司使用的技术刚出现或者刚开始流行，那么愿意尝试甚至可以用好这种技术的人，往往是善于学习、不怕麻烦的。这是技术团队里面最喜欢的一种人。

但激进的技术路线不是谁都可以玩得转的。如果想采用激进的技术路线，往往需要技术带头人是所选技术的专家，可以在技术方面辅导团队成员。同时不能让先进技术成为技术带头人的独门绝技，这样的团队是不健康的。

沉稳的技术路线

对大多数公司来说，真正的竞争优势是产品和服务，而不是采用了什么样的先进技术平台。在这样的公司里面，技术团队的任务不是提供核心竞争力，而是给公司的业务部门提供坚实可靠的技术基础，那些不能带来产品和服务体验升级的技术亮点并不重要。

对这样的团队来说，我建议采用沉稳的技术路线。简单地说，就是采用技术带头人或者团队成员都熟悉的技术，一切从实用出发。

首先，招人变得相对容易。到目前为止iOS程序员的价格仍旧是居高不下，且非常难以招到高质量的。与之相对，C#程序员、PHP程序员、Java程序员就相对便宜和好招多了。同时，可以选择的现成技术解决方案也多了，就PHP、.NET、Java等语言来说，开源的论坛、CMS系统都非常多。

对语言和平台来说，技术越来越不重要，更重要的是社区和支持。技术人员应该改变思路，不管什么语言和平台，在合适的人手里面都可以释放出巨大的力量。

以Java为例，之前是一门新兴语言，存在的最大问题是各种第三方库的匮乏。但这些年发展下来，以Apache基金会、IBM等为代表的机构，积累了大量的开源库，如Lucene家族等。

再以PHP为例，PHP早期就是以快速开发、简单易用而闻名的，在大规模计算和性能上并不占有优势，特别是脚本的运行方式带来了一些对象生命周期以及性能的问题。但作为PHP用户社区的成员，LiveJournal创造了Memcache解决了缓存生命周期，Facebook创建了HipHop for PHP大大提高了性能。

所以我们不仅可以在理论上说，没有坏的平台，只有坏的架构师，也可以从实践上说，现在几乎每个主流开发平台，都有一些世界级的优秀网站背书。而且在互联网时代，其他公司的经验很容易借鉴。

因此，采用技术带头人或者团队成员都熟悉的技术路线，积极了解社区动态，大多数公司就可以应对技术上面遇到的问题。对于乐于学习的团队而言，现在是最好的时代。



郝培强

程序员，网名Tinyfool，iApp4Me.com创始人。

注重实效的初创公司技术之路

文 / 郭欣

作为一家互联网初创公司，没有什么比快速推出产品更加重要，这意味着你可以更早把握先机，获得用户反馈，不论结果成功与否，你都必须快人一步。然而在早期的产品开发中，初创公司会遇到各种技术问题，它们或多或少地拖延了你的进度。曾有不少来自创业公司的朋友告诉我他们遇到的种种问题，其中有些问题比较普遍，我将在本文中与大家分享自己的看法。

做好独当一面的准备

作为初创公司的重要角色，技术合伙人一点都不会轻松，你必须拥有独当一面的精神准备，以及多领域的实践经验。

创业早期，你可能要亲自部署开发环境，写各种代码、设计数据库、执行测试、检查安全，当然如果还能懂一些HTML/CSS和交互设计就更好了，这会帮助产品原型的开发加速。你还得亲自选购各种服务器，寻找合适的数据中心，熟练掌握服务器的管理和维护。但这还不够，当用户规模不断增大时，你必须不断优化性能；当服务器磁盘突然损坏时你必须冷静应对并尽快将其恢复……

这听起来有些可怕，你可能认为这些工作可以聘用员工来搞定，但在创业早期，这些工作或多或少都是你不可逃避的，就连美国硅谷的不少成功创业者再次创业时，也会亲自写代码，因为这就是创业的乐趣，也是你的责任。

可又有多少人一开始就能独当一面呢？这并不是阻碍你创业的门槛，因为创业本身就是一次难得的实践，不迈出这一步永远不会获得经验。幸运的是，你所遇到的问题或许并不新鲜，通过互联网或身边的同行都可以找到答案，而你需要做的，就是做好独当一面的准备，以及始终保持强烈的学习能力。

自己组装服务器可行吗？

当然可行，其实很多初创公司都选择自己组装服务器。但正如大多数人顾虑的一样，你可能担心自己组装的服务器不靠谱，并且把它和灾难片联想到一起，这种担心是可以理解的，但我不认为有那么可怕。最近几年我开始大量使用组装服务器，它们运行十分良好。

事实上，无论是否自己组装，你都不应依赖硬件的可靠性，也不应将团队的未来寄托在任何品牌的服务器上，故障概率和组装服务器没有本质区别，而主要取决于你的应用对硬件的使用率和环境因素。而你真正应该做的是，对重要数据采取高可用保护措施，消除单点故障。比如采用RAID磁盘阵列进行冗余备份，或者在应用层编写备份脚本，甚至采用分布式存储来达到高可用性。

然而对初创公司来说，组装服务器最大的吸引力在于低廉的价格，这意味着你可以大大降低创业的成本和风险，也意味着更多人可以毫不犹豫地投入创业。购买一台品牌服务器的价钱，几乎可以买到两台同样配置的组装服务器，或者一台配置高很多的组装服务器。你还可以根据需要对服务器进行任意垂直扩展，市场上所有服务器配件的价格都向你透明。你可以使用固态硬盘SSD提升文件I/O，可以给计算密集型的服务器配置两个多核CPU，可以给数据库服务器增加到16GB内存来做缓存，而前不久1GB内存的价格也就在100元左右，128GB固态硬盘的价格也要降到千元以下，真是太划算了！

主机托管还是云计算？

在将服务器接入数据中心这个问题上，无数的初创公司都有过投石问路的经历，也付出过惨重的

代价。如今又有新的问题摆在我们眼前：我们真的不需要再去托管服务器，而将应用直接部署到云计算平台上吗？在讨论这个问题之前，我想大家应该想清楚，你希望云计算可以带给你什么，是方便的弹性扩展、具有吸引力的价格还是节省你的部署时间？当然，我希望这些方面都能具备。

如果你的应用面向海外，那么Amazon、Linode等都是不错的选择，可以帮助你在创业早期节省大量时间。国外无数的初创公司基本不需要亲自购买设备并往返于数据中心，就可以快速发布应用，这在一定程度上也促进了国外创业气氛的繁荣。但目前国内的云计算服务还有较大差距，所以在基础设施上，大多数时候我们还是少不了自己动手，主机托管仍然是多数创业公司的首选，同时还要结合自己的业务特点来设计扩展方案。

说到主机托管，数据中心的选择是很多初创公司都会遇到的问题，而这多半源于经济问题。由于国内网络基础设施的复杂现状，不同城市、不同线路的带宽，价格差距悬殊，甚至相差超过10倍。这就需要初创公司对服务进行策略性拆分，有效地利用不同品质的带宽，整体上降低成本。比如你可以在BSP带宽的机房中部署核心数据和Web服务，然后利用二线城市的大带宽部署反向代理服务器，对图片和部分页面进行加速。

Web后端使用什么编程语言性能更好？

如果你在乎对Web性能的影响，那么在选择编程语言之前，更重要的是选择平台，因为相对于编程语言来说，平台的本质决定了服务器并发的极限。如今，Linux已经成为创业公司Web后端最流行的平台，不论你的应用是计算密集型，还是I/O密集型，都可以在Linux上找到适合的解决方案，甚至还可以用它来做负载均衡，以及更多你能想到的用途。

再回到编程语言的话题，尽量选择团队熟悉并且比较流行的编程语言，因为一切都是为了帮助团队更快地推出产品，这对初创公司尤为重要。你可以选择PHP、Ruby等解释型语言，都可以在Linux平台上充分发挥性能，虽然它们在服务器开销方面仍然存在差异，可一旦需要，你只需让它们作为后端，而前端使用多级缓存，这包括内存缓存、反向代理缓存甚至CDN。



作为一家互联网初创公司，没有什么比快速推出产品更加重要

是否应该使用Web开发框架？

一套好的开发框架的确可以让开发工作事半功倍，但前提是能够熟练地驾驭它。一些流行的开源框架，比如Zend、Symfony、RoR、Django，深受开发者的推崇，但如果你觉得这些框架总是和你的需求格格不入，就没有必要强迫自己使用它。

开发框架的目的是帮你减少编写很多重复代码，基于相同的目的，我相信任何团队也都有自己的一套行之有效的开发模型，它可能来自长期实践的积累，久经考验，虽然没有那些开源框架看起来花哨，但你闭着眼睛都知道它的内部构造。初创公司奉行的是实用主义，能用就好。

而开源框架侧重的是通用性，正因如此，框架的各个层面都会引入非常多用来适应各种需要的配置选项，而这恰恰增加了整体的复杂度和学习成本，但通常大多数特性你几乎不会用到。

使用数据库还是文件？

这个问题经常会被问到，不只是来自初创公司。本质上数据库也是文件，但这里提到的文件是特指将自定义的数据结构直接写入文件。提出这样的问题大多是出于数据访问性能考虑，究竟将数据写入数据库和直接写入文件哪个更快呢？这个问题其实没有绝对的答案，而要视情况而论。

如果你要存储的是关系数据模型，显然你在读取

数据时需要支持各种查询条件，MySQL就是不错的选择，用SQL可以很轻松地实现，比如筛选、排序、分页等。在SQL的背后，数据库帮你优化查询算法，并利用已有索引，甚至还有内置缓存。同样的关系数据模型，如果用文件来自定义存储，并且需要同样的查询，试想一下，你其实是在重新实现一遍数据库的功能，而你不见得比它高效。

如果你要存储的是文档或者Key-Value数据模型，文件的确比较适合，但别忘了还有更适合的选择：可以使用MongoDB、Redis等NoSQL数据库，其中MongoDB更加强大，它使用BSON数据格式来存储文档，可以让你对直接文档内容进行筛选和遍历。这些NoSQL数据库大多都支持内存缓存，以及持久化策略，这使得一些读写操作甚至不需要磁盘I/O，比你直接读写文件当然更快。

另一方面，这些主流的关系数据库和NoSQL数据库，拥有更多普通文件不具备的特性，包括分布式访问、备份恢复、主从复制、管理工具等，你几乎没有必要自己去实现它们。总的来说，数据库的性能往往不像你想象得那么糟糕，关键在于你能否正确地选择它，并让它正确地为你工作，同时在需要时不断对它进行优化或扩容。但仍然存在一些特殊应用场景，比如搜索引擎，你需要设计一套更加高效的索引体系，而文档访问几乎是应用的核心部分，这时你花在文件访问上的时间是值得的。

为可预见的数据增长做好扩容准备

对于初创公司来说，数据的增长是令人兴奋的事情，尤其是爆发式的增长，这在互联网公司并不罕见。如果你没有对数据增长做好充分的准备，这意味着在未来的某个时间，应用性能将直线下降。更麻烦的是，为了重构数据存储方案，并且迁移数据，你可能要忍痛停止部分服务，我们都不希望这种事情发生在初创公司。

从一开始就尽量让你的数据存储支持分布式和水平扩展，以达到动态扩容的目的。如果你使用MySQL等关系数据库，那么尽早让应用支持分库及分表策略，这也意味着可以很轻松地将数据分布到不同的数据库服务器上，最终让数据表的记录数不要超过百万。而对于NoSQL数据库，你只需要尽早做好数据扩容的规划，并且让应用支持这种策略。不要将这些看成是多余的工作，它们没有你想象得那么复杂，你只需要稍微费力气，就可以在未来获得巨大的回报，何乐而不为呢?

尽早使用代码版本管理

这虽然不是一个技术问题，却是初创公司不可忽视的问题。一旦项目创建，从第一行代码开始，你就应该引入代码版本管理，不论是SVN还是Git，用你熟悉的就好。尽早使用代码管理的好处非常多，它可以让你的代码超越时间和空间的限制，这对初创公司尤为重要。

超越时间的限制意味着代码可以更加安全地存在，你不用担心本地磁盘崩溃或者误删代码，因为你总是可以将它恢复到历史版本，而代码对初创公司尤为重要。当然别忘了对代码库进行备份。

超越空间的限制意味着团队协作开发将不存在任何困难，不论团队成员在哪儿，只要将代码库部署在大家可以访问到的地方，大家就像近在眼前，每个成员对代码的操作一目了然，因此快速协作也是初创公司的制胜法宝。

代码版本管理的意义不只这些，你还可以使用它快速部署应用，这对早期的产品迭代更新非常有帮助，你可以将代码库放到产品服务器可以访问到的地方，使用代码更新功能直接更新代码。如果发现应用更新后存在严重Bug，你还可以将应用立即恢复到上一个版本的状态，而这也只需要一个命令即可。

另外，你还可以将代码库集成到一些开源的Bug跟踪系统，比如redmine、trac等，并稍加配置，当你签入代码时，它们可以自动将产品特性或Bug修复与代码文件进行关联，这使得整个创业团队可以更加高效地协作，始终保持一致的方向。



郭欣

监控宝创始人，《构建高性能Web站点》作者。新浪微博：@guoxin

技术人员创业与融资的八大误区

文 / 洪峥

科技型创新企业在融资上往往不顺利，一般有内外两种主要因素。本文归纳了八个易犯的错误，分析其中原因，与青年创业者分享。

我曾写过一篇名为《高科技？想要爱你不容易》的博客：

从事高科技行业的人常有被炫目光环笼罩之感。其实，深究高科技行业的现状及本质，却发现不利因素颇多，可称为“内伤外伤”。“外伤”是指融资环境，资本（钱）追捧最多的是房地产、金融等暴利领域，现实中人们并没有看到资本（钱）向高科技倾心多少。胡润排行榜上的中国超级富豪许多是搞房地产的，没见到许多搞高科技的，这便是证明。“内伤”是指从事高科技的人创办企业，其实他的素质能力未见得如他自我感觉那么高。一般来说，科技出身的人一不懂资本，二不懂人，往往还自以为是（我也是科技出身，无意贬低自己）。而这几条是做企业经商的大忌。高科技公司创业者大多艰难坎坷，融资困难，都与此有关。投资方遇到此类自高自大的科技专家，往往敬而远之，心里清楚难以合作长久。外伤+内伤=雪上加霜。高科技？想要爱你不容易。

误区一：自恃精英、估值过高

我在1999年曾参与创办一家网络企业，若干年后，一位当时公司研发部的骨干来找我。他说他创办了自己的小型科技公司，打算开发网络安全监控软件，希望我帮他融资。当问起投资额和回报时，他希望投资100万美元，占其公司20%的股权。他的想法在青年科技创业者中很普遍：需要足够的钱研发产品，并初步打进市场。但他有一点没有想到：他提出“100万美元，占他公司20%的股权”实际上就是在告诉投资方：他的公司现在价值500万美元！那么他需要向投资方证明凭什么他的公司现在值500万美元。当时的状况是：他花了几十万元人民币注册了公司，人员仅他一人，还有一个朋友兼职帮忙，产品还没研发出来，市场营销进账一分钱没有。他在这个阶段应该做的不是寻求风险投资，而是继续筹备，试探市场，以亲朋好友资助及政府创业扶持资金等方式解决种子资金。

误区二：同床异梦、各怀心事

如果把投融资对接的平台称作“床”，那么融资的技术型创业者和投资方各自的心事完全不同。技术型创业者擅长的是技术，所以滔滔不绝地谈技术；投资方想的就是一个字“钱”，盘算如何低投入高回报。因此，创业者兴致勃勃谈的技术细节，投资方却一点都不关心。前文提到科技出身的人“一不懂资本，二不懂人”，在路演时就暴露无遗。要知道，从技术到成型的产品再到市场获得回报，long way to go，往往是十万八千里，其中充满了不确定因素。曾经在中科院计算所听一位网络技术专家路演，不要说投资方听不懂技术细节，连在场的计算所领导都说：“我听了六次，还是不懂。”对此，我的建议是：不谈技术细节，只谈什么客户在使用这个技术、可能付多少钱。

误区三：市场太大、定位模糊

在项目路演时，常听到技术型创业者介绍市场：“中国有13亿人口，手机用户数为世界第一……”似乎他的移动互联网项目拥有巨大的市场空间和无数的客户。其实，“中国有13亿人口”，可能和你的项目一点儿关系都没有；“手机用户数为世界第一”，可能没有一个手机用户参与你的项目！技术型创业者并没想清楚他的细分市场是什么，到底什么客户在使用或将要使用他的产品服务，然后计算出切实的市场空间和份额。技术型创业

者一定要明确具体和独特的定位，否则无法应对科技产业的残酷竞争。模糊的定位是无法让投资方掏钱的。

误区四：财务预测？财务实收？

在项目路演时，常听到技术型创业者介绍漂亮的营销财务增长曲线，从现在起，逐年递增。投资方知道那些都是预计虚构的。技术型创业者必须注意提供过去2~3年的财务数据，哪怕是亏损，因为过去的财务实收是今后财务预测增长的主要依据。投资方不是傻子，融资时能够提供过去财务数据的创业者比只能展望未来的人具有更大的优势来说服投资方出钱，因为投资方这时可以估算他的回报和期限。

误区五：复制抄袭、缺乏独特

由于IT互联网、移动通信的特点，复制抄袭项目或技术极其便利，例如团购和社交网络。我在中科院研究生院MBA授课给学生项目演讲，或者在其他场合做科技型企业融资路演时，经常发生项目雷同“撞车”的事。例如：3D视图或动画房屋装修或售房；团购和电子商务的中介服务；利用GPS定位或RFID协助搭帮旅游、找餐馆、购物等，这些项目的思路相同甚至产品雷同。如果技术型创业者思维懒惰、投机取巧，不肯挖掘独特的思路、模式、产品和服务，那么我劝你与其从事科技行业还不如从事传统行业。你模仿他人的成功开一个餐馆，完全可能有客户，你模仿他人科技项目搞一个团购或中国版Facebook，却可能一个客户都没有，因为互联网客户没有忠诚。

误区六：只谈索取、不谈回报

这是一个技术型创业者常犯的错误：他总想着自己的技术多么好，别人早就该投钱了。一张嘴就是“你投资方有没有钱？别总是尽职调查耽误工夫！”他的思维是单赢不是双赢，没有充分考虑投资方的回报。这犯了大忌，而且很不公平。因为投资方投入的是钱，如果项目失败将血本无归，而技术型创业者主要投入技术，即使项目搞一段时间失败了，他的技术不但没有损失，反而在这一段时间借助投资方的资金而发展得更成熟了。我见过这样一段经历：一个技术型创业者借助一个又一个投资人的资金，完善了自己的技术，而投资人全都损失惨重，他在投融资圈子里的名声由此也就坏了。

误区七：技术超强、水分过大

这是常见现象，技术型创业者路演时往往声称：某项技术是国内外领先。其实，许多都经不起推敲。尤其是对于具有国内外背景的投资方，他们都知道两种情况：1. 国内环境压力令创业者浮躁，注重眼前利益，无心无力潜心科研；2. 国内在图书馆等方面与发达国家仍有差距，不利于科研人员广泛深入了解世界尖端技术（还甭提美国和西方国家对中国的高科技封锁）。往往是技术型创业者误以为自己的技术世界领先，其实是缺少渠道而不知道还有比他先进的。我的建议是：与其轻易地宣传技术超强（让人误解水分大），不如向投资方说明：该项技术已经赢得了多少客户，赚了多少钱。

误区八：兼职创业、风投大忌

这是风险投资公司很忌讳的事：技术型创业者告诉风投，核心团队成员暂时还在原单位上班，一旦条件成熟，资金到位，就会辞职全力投入新公司创业。投资方想：“你们是脚踩两只船，用我的钱，办你们的事。成了，你们赚钱，失败了，我赔钱，你们还在原公司上班挣工资，左右逢源。”大家要知道，创业极其艰难，即使破釜沉舟全力以赴，也失败多多，何况你三心二意呢。投资方想听到的是“我们核心团队全体成员全都从原单位毅然辞职，义无反顾地投入新项目！”这样显出双方都在承担风险，显出技术方的合作诚意，投资方愿意出钱。



洪峥

加拿大瑞普公司董事，北京互联星空公司高级投融资顾问，美国秦通公司创始人之一，中科院研究生院MBA导师教授，北大、清华客座教授，畅销书《大国民企正道》作者。

“解决遗憾”是极好的创业契机

文 / 郭子威

郭子威，网名纯银，经历传奇。1996年考入成都市检察院，做司法警察。2000年从检察院辞职，进入电脑商情报社，任《数字化用户》杂志执行主编、《游戏天地》周报副主编。2004年，受投资人邀请去福州创建“天下创世公司”。2006年已有盈利，从VC那里拿到了数百万投资。2006年底辞职加入网易，2008年初从媒体转型产品，在门户下面新组建产品部；2009年调动到网易杭州研究院，任相册产品部总监。目前，郭子威已从网易辞职，处于创业准备阶段，他在本文中分享了自己对创业选择的思考。

36岁，离职，计划二次创业。

上一次创业在福州，苦战两年未达到个人目标，无奈离开，加入网易去做一些“有影响力的事情”。又过了五年，经验值Level UP，影响力却仍是水中风月。无奈离开，再次踏上创业之路……

作为第二次创业，给自己制定了两个前提：

- 创业是为了追随内心的冲动，这是唯一的理由；
- 无论怎么努力，创业成功都是偶然，“享受创业状态”比“死磕功成名就”更重要，创业只是因为我喜欢过这样的日子，仅此而已。

然后我便开始寻找创业的方向、主题、项目。

从兴趣出发

我打算从自己的兴趣爱好出发。如果不是发自内心的热爱，必不能保持多年的关注；如果没有多年关注，则很难发现隐藏在不起眼处的需求空档。灵感来自于你的生活方式，由它决定知识结构、人脉关系以及对市场和用户的理解。这些东西很难靠恶补或抄袭获得，而是发源自你身处的环境与生活习惯。

你最爱好的与最常接触的领域，最容易迸射灵感，挖掘机会。换个角度看，在熟悉领域内，自己就是典型用户，甚至是专家用户，虽有以己推人的隐忧，至少避免了凭空臆测——后者的失败概率是前者的10倍之多。

基于这个出发点，在排除若干个自己并非典型用户的创意后，下决心将“旅行”作为产品方向。像我等IT宅男，除了旅行，就没什么付诸实践的爱好了。2011年春节在柬埔寨旅行11天，期间有不少遗憾，**“解决自己的遗憾”往往是极好的创业契机。**

最近一年，国内旅行新产品扎堆发布，粗略估计也有30款以上，但无一款崭露头角。一方面是因为这爱好如此大路货，而IT宅男们的生活雷同，既然从爱好出发最佳，自然容易撞衫。另一方面又因为旅行的低频度，很难形成有黏度的用户群，对UGC极为不利。

这就涉及产品突破口的选择。

两大误解

按照我的理解，用户价值第一，商业价值第二。虽然这个观点早就滥大街了，但在UGC产品中却衍生出不少误解。

误解一：鸡蛋悖论

轻视获取第一批优质用户的难度，假设（意淫）用户天然就喜欢来此发表内容，以此为依据来设计产品架构与功能。怎样才能吸引读者？因为有很多优质的内容。怎样才能吸引优质的内容？因为能够为作者带来读者的热情互动。这难道不是一个死循环吗？从0到1恰恰最是艰险。鸡生蛋，蛋生鸡，无穷匮也……阁下做梦。

误解二：直奔主题

既然是UGC，自然以内容为本。可用户为什么要使用你的产品，在这里发表内容？往往源于他们对品



在郭子威眼中，创业的唯一理由是为了追随自己内心的冲动

牌的好感。有时为了“攒好感”可能会迂回前进，并不直接从UGC做起，而是先做另一些增强用户信任感的事情。这确实绕了弯路，但至少种下的不是无根之花。反倒是死活要开门见山UGC的产品，常常让用户找不出一个爱它的理由。

归根结底，新产品或新版本给前100个优质（活跃）用户提供的满足感？正是它的立身之本。直接穿越去幻想第1千个、1万个活跃用户会得到什么样的满足感，这样的产品多半是要仆街的。不能解决前100个优质活跃用户的问题，其后的产品架构都是沙地上的大厦，云上的城堡。

笨拙地追求用户价值，精明地追求商业模式，在互联网上前者更容易存活下来。

需要有说服力的商业模式

回归到旅行主题，既然旅行的频度低，用户黏性自然也低，我对产品突破口的定义是“工具化”，从旅行相关的工具入手，提供立竿见影的帮助。在实际案例中，旅行UGC产品佼佼者的蚂蜂窝以精心制作“路书”发家【注：路书就像一本流水账。里面记载的是：什么时间到什么地方、沿途的风光、住宿的安排、费用的支出等】，穷游网初期则提供出国游的咨询交流，均顺利起步。这不是绕远路和抄近路的区别，而是活路和死路的区别。

只不过，用力维护用户价值，与寻找商业模式并不矛盾。我们都相信，能创造用户价值的产品一定能找到自己的商业价值，但商业模式决定了盈利预期，盈利预期决定了投资大小，投资额减掉运营成本，则是你的生存时间。谁都不想烧光融资轰然坍塌对不对？

所以，上策是距离现金流尽可能地近一点，早一点自己养活自己；中策是预先设想一套可靠的盈利模式，说服VC拿到更多的融资；下策则是用力地压缩成本，给自己省出时间来积累用户，创造机遇。

做产品是需要理想主义的，可惜“不计得失”、“放手一搏”的理想主义者既不多得，也不容易聚在一起。积极的前景才能鼓舞同伴、凝聚同伴，从这个层面上来讲，持续而有力的数值增长、有说服力的盈利模式、足够的现金储备，都能够给团队信心，而不是奢谈理想，奢谈梦想。

作为一个诚恳的建议，不断寻找并不背离用户价值的、适合自己的商业模式——从开始就得花费偌大的心思。就像一位投资人告诫我说：“我不担心你的产品不好，我担心的是你在拿到下一轮投资之前就花光了我们给你的几百万，青黄不接，熬不下去。”这句话是金玉良言。单单追求用户价值的话，除非数值成长特别快，或是研发运营成本特别低，否则很容易遇险。比如你的用户价值远不到变现的程度，钱没了，人心也散了。何以为继？

以上是我对创业选择的思考逻辑：从自己爱好的领域入手，追求立竿见影的用户价值，尽量靠近某一种讲得通的商业模式。由此找到匹配的项目，则全力投入，找不到则暂且作壁上观，蓄势待发。硬件软件/商业生态/产品理念等的进化，不断制造“满足这个那个需求”的机遇。大部分创新并非凭空构思，而是善于组合默默变化中的市场要素。昨日天堑或许明天就被时代填平。

定义中国轻博客

点点网创始人许朝军专访

记者 / 陈博

许朝军曾于2000-2005年供职于搜狐，2006-2009年期间担任千橡集团副总裁兼校内网负责人。2009年11月从千橡离职，2010年1月12日，加盟盛大网络并任职首席运营官。2011年2月创办点点网，投身于移动互联网SNS创业浪潮。同年4月，点点网公测上线。目前，点点网注册用户超过500万。许朝军在接受《程序员》采访时说，早在大学，他就已在心中播种下了创业梦想，十年过去了，这颗梦想的种子开始生根发芽。

创业：风险与机遇并存

《程序员》：谈谈你从技术人员到创业者的历程吧。

许朝军：1996年，Java进入中国，那时我还在清华读书，中国第一本介绍Java的书就是由我们学校的教研室编写的，所以我最早接触的是Java语言。随后我开始在中关村一些公司兼职，做了一些项目，再后来我开始用Visual C++，做Windows系统上的客户端程序开发。

我在毕业后去了搜狐，做即时聊天工具客户端——“搜Q”，然后参与了一款浏览器的开发。我们的浏览器做了一个类似美国在线的模式，为用户提供拨号上网服务，当时我们在全中国也是第一家做浏览器的公司。

那时的程序员基本上都是前端和后端兼顾的，有点像现在的美国。做过很多项目后，我就开始逐渐从技术人员向产品人员转型，“搜Q”、浏览器等几个项目，大部分代码都是我写的，同时我还需要思考如何打造一款产品。

2006年，我看到了Web 2.0的机会，陈一舟对Web 2.0也非常感兴趣，所以我去了千橡，那时我已经基本上不写代码了，而是作为一个产品经理的角色，开始管理产品、技术和市场运营。整个2010年，我去盛大待了一年。那时我开始思考，自己的职业生涯基本上到一个里程碑了，希望能够有机会完成自己的创业梦想。

《程序员》：你为什么选择在2011年开始创业？

许朝军：我在读大学时，心中就埋下一颗种子，想成为创业家，我觉得人生一定要有一次创业经历，否则我会非常遗憾。但我并没有选择在毕业后就投身创业，作为一名技术人员，我十分谨慎。一名技术人员，在技术做好的情况下，要能延伸到产品、运营、市场，而作为一名CEO，还需要具有资本运作的能力。以前做事，都是从部门的角度去看问题，但现在，要学会管理人事、财务、行政。与之前的经历相比，创业是新的挑战。所以在十年之后，当我认为创业的各方面风险都降到很低时，才决定开始创业。

创立：用互联网精神去做点点

《程序员》：你为何选择将轻博客作为创业的切入点呢？

许朝军：2005年，我看到了Facebook的机会，它的实名制策略让我坚信它可以成功，所以我在千橡时也开始探索这个方向，并获得了很大成功。但我明白，在我创业时，再做Facebook型社交网络，是没有任何机会的。一个人把事情做成了，并不能代表他再做相同的事还能成功，因为机遇窗口已经关闭。但整个社交网络的机遇并没有关闭，互联网产品的成功源于它能够满足用户需求，而用户需求是一直存在的，当我看到Tumblr时，我感觉找到了新方向。

每个人对创业有自己的目标，有些人为了赚钱而



许朝军说，专注地把一件简单的小事做到极致就是伟大

创业，但这种公司很难长大。因为互联网法则是：要先有价值，才能赚钱。

互联网在中国发展了十几年，现在中国的互联网仍缺少品牌，不注重用户体验，我认为这些都是可以改变的。我希望“点点”首先能成为中国互联网的原创中心，目前我们的互联网太缺乏原创魅力了，而中国的摄影师、插画师、作家、设计师、记者都能产生高质量的内容，所以我希望能够基于这点，打造基于兴趣的社交图谱，从商业模式上看，“点点”是个兴趣推荐引擎。

《程序员》：有媒体将你定义为中国的轻博客之父，轻博客是什么？点点具备了哪些特色？与目前如日中天的SNS和微博相比有怎样的区别？

许朝军：可以说点点网是中国第一个真正意义上的轻博客，是将Tumblr模式在中国落地的本土化发展，更依据对中国互联网用户和创业环境的深度洞察与了解，将产品功能与用户习惯紧密结合。从创立初期，我们在中国市场上有过摸索，有过无数的困难，也走过弯路，但我们很快就调整过来了，发展上逐渐清晰，一句话概括点点网就是：最简单好用的博客、社交化的博客、移动化的博客、开放性的博客，在点点网上用户信息交互产生人际交互。

目前市场上SNS和微博都是基于人际交互产生信息交互的产品，一切内容的交互与传播轨迹都是基于人际交互而产生。点点所代表的轻博客平台，人际交互是辅助功能，信息的生产、发布与传播基于兴趣产生关系网络和社交，并将传统博客产品进行了质变和升级，革新的功能中不仅让用户发布内容和展示内容更加简单便捷，并融入了社交化的功能，同时多终端移动化互通，全平台开放。此外，坚定地将互联网的开放性和平等性做到极致，让平台化、平行化的特点凸显到极致。

《程序员》：你选择了与创新工场合作开始创业，谈谈其中的原因吧。

许朝军：李开复在Google时，我还在千橡，人人网与“开复学生网”有很多合作，我们就这样成了朋友。开复创立创新工场后，也看到了移动互联网下SNS的新机遇，所以他邀请我创业。对于我来说，当时有几个选择：第一，可以找VC，而在2010

年时，VC真正看到移动互联网SNS机会的并不多；第二，自己出钱，但需要自己找房子装修、雇人……这样太耗费精力。而开复能够帮我解决这些问题，让我专注于产品，并且创新工场能够帮助我在创业第一天就能引入现代化的、硅谷式的企业架构和资本结构。可以说，在点点网的创业初期，创新工场对点点网的孵化和推向市场起到了一定的促进作用。

2011年4月28日，点点网正式完成A轮超过1000万美元的融资，由联创策源、红杉资本以及创新工场投资。后来点点网独立成长壮大，在市场上快速成为轻博客领域独立的品牌和一面旗帜。

《程序员》：点点注册用户数在2011年6月3日突破100万，距开放注册仅63天，平均每分钟增长11个注册用户。相较于同等量级的用户积累，Twitter用了一年多，而Facebook用了10个月。以5倍于Facebook的速度增长，你有没有总结其中的原因？

许朝军：“点点”在初期是封闭注册，当时大家都听说我在创业，但都不知道到底是什么项目，这让“点点”充满了神秘感。而“李开复”、“许朝军”、“创新工场”、“Tumblr”这些关键字，激发着大家的想象，我们让外界的这种需求压抑了两个月，所以当点点在2011年6月开放时，大批互联网从业者涌入，为“点点”带来了业界影响力，带来了“点点”初期时用户数量的高速成长。

《程序员》：能分享下点点网站架构的升级历程吗？

许朝军：最初，我们选择了大家都较熟悉的、成熟的Java架构，并采用了非常简单的Web+DB模式。随着我们的开放注册，用户数到达10万量级时，我们对一些服务进行拆分，引入了Redis、JMS。2011年4月后，点点开始在市场上进行推广，这时流量增长非常快，为服务器带来了很大压力。

我们迅速地将Web Service、服务、存储等，全部进行分布式处理。流量增加后，我们就可以用加服务器这种最简单的方式，把呈指数级增加的流量压力承担下来。在推广后期，流量趋于平稳，就可以让部分服务器去实验一些新技术，达到动态的调配。我们现在在想下一个架构，在下一个指数级上如何去调整新的分发策略、通知策略。

创造：把小事做到极致就是伟大

《程序员》：作为一家创业公司，你是否有想过控制团队的规模？

许朝军：刚开始创业时，我和两位以前的同事，作为公司的联合创始人，并找来了三个技术人员，两个产品人员，组成了八人的小团队，花了两个月时间，就把点点做出来了。现在团队人数在50人以内，我希望把这个人数保持很长一段时间，因为这样规模的团队更容易专注地把一个小而简单的事情做到极致。我非常享受现在的状态，公司没有太多的管理制度，人际关系也很简单，大家非常专注地做每件事，并能根据用户反馈快速迭代。

《程序员》：Facebook的上市给你哪些启发？

许朝军：Facebook的上市可以用“水到渠成”来形容。首先，Facebook创业用了8年时间，是时候给投资人和员工兑现当初创业的承诺了；其次，上市能让Facebook成为公共性质的公司，有更多的行为约束和更广阔的舞台；再次，能获得更多的资本和资源去与其他巨头们竞争。

《程序员》：在你看来，成功的互联网产品最需要具备的品质是什么？

许朝军：最重要的是要不停地用技术和产品创新提高自己的产品价值，酒香就怕巷子深，因为信息不对称，这个市场很大，行业竞争激烈，所以一定要让价值在用户那里显性化。做互联网，就要相信互联网精神，互联网最重要是TCP/IP协议，它代表的是对等和开放，这是互联网的哲学。我们需要用互联网精神去做互联网。

精实起步
糗事百科创始人王坚专访

记者 / 林溪

在王坚看来，互联网服务就是一个个混合生命体，糅合了人类的工作和自动化系统的工作。从这个视角来看，糗事百科是非常有潜力的一个小生命，它靠自己的努力从一个兴趣项目成长为了一个职业项目，王坚很享受和成长的过程。

前期准备必不可少

《程序员》：你最初创建糗事百科时，只是以个人网站的形式供网友分享一些各自的趣事。后来怎么想到以此为基础来创业的呢？你做《秘密》和《起床大作战》的切入点分别在哪儿？

王坚：糗事百科创办自2005年，最早是作为兴趣爱好而创建的网站，我经常用它测试一些产品和运营上的想法。后来，我的大学同学高志接手了网站的运营，提升了运作效率，把它过去积攒的用户规模释放出来了（之前因为打开速度太慢而阻挡了很多用户的访问，改进后用户都能流畅访问了）。很快，流量就翻了几倍，同时日常维护工作量也越来越大，需要全身心投入，于是我就把它作为全职项目，开始创业了。

因为创业初期大家对糗事百科的界面设计没有达成共识，我们就先做了《秘密》练手。《秘密》是以前就有的项目，重新改版成了类似Quora的模式，并且尝试了国际市场；《起床大作战》也是基于兴趣的产品，后来证明它还是有实际价值的。如果重来一次，我们会第一时间推出糗事百科的App，并专注于此，而《秘密》和《起床大作战》可能至今还未面市。

《程序员》：当你准备投身创业时，做了哪些前期准备？

王坚：最大的准备就是把产品做好，达到一定规模，这样自己就有信心了。接下来的难点是说服我的家人和创业伙伴的家人——本来都有不错的工作和稳定的收入，突然要跳到一条不知道什么时候就可能沉的船上，收入也会下降一阵子，这难免让家人担心。虽然基于数据分析，让我们对糗事百科的发展很有信心，但这种信心要传递给家人却很困难，需要做很多工作。妻子支持我创业后，建议我们等三个月再辞职，一方面能攒点钱注册公司，另一方面可以多认识一些创业圈的朋友多学习。这个建议让我们过渡得非常从容。

《程序员》：在你看来，一位优秀的产品/技术人员与一名优秀的创始人之间有哪些差距？

王坚：在企业内部工作，可以专注于产品本身，身兼管理岗位时，只要带着大家一起干活就好，企业HR解决了绝大部分的人员问题。作为创始人，考虑团队建设比考虑产品更重要，因为再好的想法没有执行也是零。如何找人、如何留人、如何平衡、如何升级，这些都是不得不去面对的问题，解决好它们，才有可能做成事情。优秀的产品人员或技术人员，不需要去面对这些问题，也没

有锻炼的平台。

《程序员》：在构架糗事百科时，你是如何选择网站的架构、数据库和语言的？

王坚：糗事百科的用户行为非常简单，所以我们没有花多少时间在架构上，而是从动态的视角来看，能够满足每个发展阶段的用户需求就好。糗事百科的第一个版本是Sam用PHP花了一周的业余时间写出来的，后来我的同学Cgroom花了两周用Ruby on Rails改写了一遍，再后来ShiningRay用了一个月升级到了Rails 2.0。由于我们不擅长优化Rails的性能，所以在Rails 2.0暴露出一些性能问题后，就让高志接手，他又花了一个月迁移到Tornado。

整个过程没有深谋远虑，也没有超前设计，都是遇到问题后用最顺手的方式改进的。如果有一天糗事百科达到日一亿PV，架构设计才会成为一个重要的问题。

数据库一直用的是最常见的MySQL，语言倾向于代码密度高的Ruby和Python，因为它们能够提升编写和维护的效率。

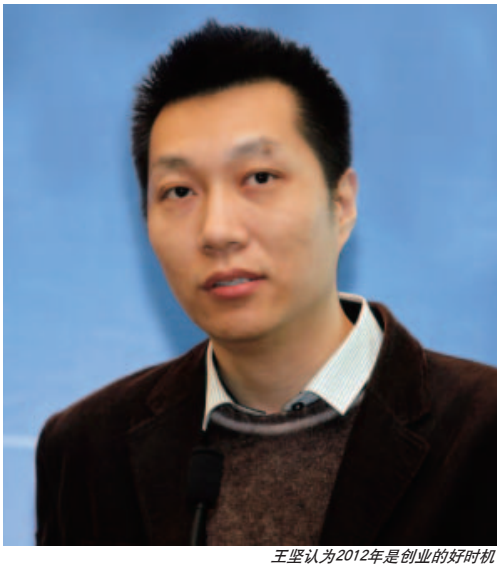
制定协议帮助解决团队矛盾

《程序员》：创业至今，主要遇到了哪些困难？

王坚：最大的困难是沟通。我们几个人由于各种机缘巧合从天南海北凑到了一起，大家的价值观并不统一，过去的从业经历和处理事情的方法差异也很大，也没有太多时间做深入的沟通就开始干活了，导致很多事情看起来在推进，但实际上并不是基于有效沟通后达成的共识，只是沟通不到位的相互妥协。

我们成立公司时就想做糗事百科的App，可晚了半年才推出，也是基于这个原因。我们花费了半年以上的时间初步解决了沟通问题，方法就是允许大家犯错，然后指出错误，让每个人都清楚地意识到自己行为的后果。这不是一个非常好的办法，代价和风险都非常高。

《程序员》：你的创业伙伴们对你提供了哪些帮助？



王坚认为2012年是创业的好时机

王坚：目前我们团队有六个人，每个人都承担很多工作，比如iOS产品设计、开发、推广外加一些网站前端的工作都是一个同事负责的，大家凑到一起才有了现在的稳定快速跨平台的糗事百科。因为我不熟悉开发，在全职创业之前是和Sam、cgroom、ShiningRay合作运营糗事百科的，没有他们的帮助也没有糗事百科的今天。

因为糗事百科是一个用户分享糗事的平台，用户也是我们的伙伴，没有他们的呼声，糗事百科可能早就关闭了，没有他们的热心建议，糗事百科也不会有这么多人性化的细节。

《程序员》：有段时间，网上盛传的“糗事百科团队股权纠纷”对你们造成了很大影响。从这件事中，你有哪些经验可以与大家分享？

王坚：之前霜叶用了很长的篇幅描述所谓的纠纷，其实里面包含了大量与事实不符的情节。比如他说给了我们一份一年之内让用户规模增长20倍的发展计划，促成了我们的快速发展。经过在知乎上的对质，实际上并没有这样一份计划，我们在一段时间内具体增长了多少倍、通过哪些方法实现了增长，他也并不了解，类似的杜撰情节还有很多。

事情的本质很简单：我和高志与他沟通过几次全职创业的事情，他拒绝全职加入的邀请。

如果我欠霜叶人民币若干而不还钱，这是人品和诚信的问题，但我和我的公司并不欠钱，因为股

份和欠款完全是两个概念。注册公司时所确立的各股东的股份是伴随vesting协议的，每工作满一年，可以兑现自己股份的25%，四年兑完。

业界之所以产生这样的规则，主要因为创业是一个长期而艰难的过程，达到一个比较稳固的状态通常要四年以上，如果有股东半路走人就会对其他股东非常不公平。

比如我干了两个月就走了，同时拿走了公司一大把股份，其余股东没日没夜熬出来的工作成绩我都可以坐享其成。他们肯定会想办法把公司关掉另起炉灶，我得不到什么，他们也会浪费很多不必要的时间和精力，两败俱伤，而这个局面完全可以通过vesting协议预防。vesting的概念我们也和霜叶沟通很多次，最后是这样的结果我们也很无奈。

希望这次事件能让大家更注重vesting这个常识。我们目前运作还不到一年，每个人手里的股份都还是0，我们都在努力工作以兑现几个月之后的第一个25%。

2012年是创业的好时机

《程序员》：据说去年糗事百科的估值已达3000万元人民币，月收入10万元以上，已经具备成功商业运作和融资发展的基础。你对此有何看法？

王坚：没有人用这个估值和我们谈过收购或投资，这个没有真实市场交易的估值我们也不清楚是如何开始流传的。我们暂时看不到糗事百科可以发展到上市规模、大家的股份可以变现金的那一天。但基于现在的用户规模和糗事百科的黏性，我们有坚定的信心让我们团队成员在不久的将来都过上不错的生活。

《程序员》：目前你与哪些投资公司或投资人有过接触？是否考虑融资？

王坚：我们接触过王啸团队、经纬创投、晨兴创投、执信资本等几家，他们都给过我们非常有价值的建议，但我们看不到自己能达到上市的规模，如果拿了投资，资方无法获得回报，所以还是继续走lean startup的路线稳当，连天使投资都没拿过。但无论拿不拿投资，经常和资本圈内见多识广的高手聊聊，成为朋友，绝不是坏事。

《程序员》：移动互联网带来了哪些新的机遇？

王坚：每个人手里都有一个可编程、可联网带有诸多传感器的设备，这件事情太惊人了。单一功能的手持设备将逐渐被取代，变成纯软件（如iPod、PSP、快译通、导航）或软件+传感器（比如Square），以前因为成本问题没有出现的信息化设备将会出现，以前无法想象，每个人能手持一个设备，扫条码查价比价的。人类社会的信息化程度将迈上一个新的台阶，互联网上可以交换的信息自然也增多了很多倍。并且，手机在融合音乐播放器、视频播放器、PDA之后将会继续融合掌上游戏机，凡是能够放进口袋的电子商品，都会被手机吸收进市场。手机将会变成数字消费品（音乐、视频、游戏）和数字应用的消费终端，其中数字消费品的份额可能很大，而数字应用的领域可能会出现平台。

和传统软件产品相比，移动应用主要面向碎片时间，必须快速启动，快速完成任务。所以我认为最具有挑战性的是如何实现快速地验证用户需求。虽然移动应用开发周期较短，但移动产品从设计到实现的时间还远不及Web产品。

《程序员》：为准备今年投身创业的朋友们分享点建议吧。

王坚：首先，要想清楚自己能为社会解决什么问题，以及为什么自己比其他人更有优势，不要因为工作不顺心或认为创业看起来很酷而创业。

其次，如果能快速赚钱养活团队，就有了“眼”，“气”就长了。我做过一个计算，一个站点达到日5.5万独立访客，大约能产生5000元/月的广告收入，可以够一个人的日常开销外加带宽等基本开销，如果能快速做到日5.5万独立访客/员工，“眼”就算做好了。

移动设备和社交平台的发展非常激动人心，如果你想做的不是纯移动应用或社交应用，也一定不要忽视这两大领域，最好把它们融入自己的基因。今年几大公司都在减员，这意味着很多有经验的人会从大公司里出来，人才垄断的局面开始被打破，这是难得的好机会。

LBS，在探索中前行

《街旁》CEO刘大卫专访

记者 / 杨爽

国内LBS的盈利模式一直不清晰，变现能力长期遭到质疑；新型的LBS交友应用冲击传统市场，老牌LBS应用该如何应对；微博、微信等社交新势力的位置服务逐渐成熟，单纯提供位置服务的LBS应用的生路在哪里……带着这些疑问，《程序员》杂志专访了《街旁》CEO刘大卫。

团队建设有点难

《程序员》：《街旁》现在的规模有多大？创业初期扩展团队主要基于朋友关系，这带来了哪些好处？

刘大卫：现在团队约有100人。一开始主要依靠朋友关系扩展团队，其好处是易于保持团队文化。团队创建之初，成员基本上是20几岁的年轻人。随着公司的成长和发展，我们需要一些更成熟、水平更高的成员加入。而怎样去招一些有经验的员工及这些员工如何与现有的年轻团队融合，是挺大的挑战。

《程序员》：《街旁》是怎样应对这一挑战的？

刘大卫：招聘工作现在由人力资源团队负责。主要在招聘HTML5、iPhone和Android等技术上比较强的人才，最好还能有相关方面的创业经历。架构方面，希望能吸引一些有大公司工作经验的人加入。

创业公司很难像大公司那样提供稳定的工作环境和高工资，但还有很多人愿意加入，显然是因为这里有其他更吸引他的东西。例如，在这里，他自己设计的UI、编写的代码，不但容易被公司采用，而且会对公司的下一步发展产生很大影响。而这在大公司里往往很难实现。

以前，只是一些朋友在一起做项目，而现在需要融入公司的工作模式和流程，这是我第一次面对的。每当有新成员加入，一般都会带来一些新的观念和方法，可帮助我们用更全面、更专业的标准来评估我们的产品。

《程序员》：《街旁》的员工构成比例是什么样的？

刘大卫：我们的人员分布比较平均，技术研发团队、运营团队、推广与销售团队大约分别占1/3。在下个阶段，研发团队的比重会加大。

用户价值带来用户黏度

《程序员》：同类产品非常多，《街旁》如何做到与众不同？

刘大卫：从2009年到现在，LBS变成一个行业话题，经常被讨论。之所以会有这么多LBS App，是因为大家都没有想清楚最好的模式，都处于摸索阶段。但每个细分领域都有很大的发展潜力，哪种服务或哪种技术能发展起来，目前还很难说。

很多人在炒作LBS和O2O商业模式，而我们的观点是先赚人气再赚钱，因此更关注给用户提供的体验和服务是什么？该怎样服务用户习惯？一个人到了某个地方，想做个分享，晒晒他正在哪里，在做些什么，这是很重要的用户习惯。这一年半以来，我们积累了一些用户习惯，考虑在下个版本中将一些比较细致的用户体验做好。例如，当用户拍一些美食、风景或人像的照片做分享时，可以用一些技术让图像看起来更漂亮。


刘大卫说，社交产品的成长要依靠口碑

《程序员》：《街旁》在推广方面做了哪些工作？收效最大的是哪种？

刘大卫：我们一直都在努力将新浪微博、人人和豆瓣等SNS的同步功能和体验做好，方便用户将信息分享到它们上面。用户可能在《街旁》上没有朋友，但通过SNS同步功能可将《街旁》上的信息同步到有自己朋友圈的SNS上。在《街旁》上线初期，这带来了60%~70%的新用户增长。

社交产品的成长依靠口碑，SNS推广就是一种线上的口碑营销。创业初期，《街旁》将用户群划分得很细，甚至细化到只有七八十人的小群体，意在为这些人提供好的服务和体验。虽然现在用户量比以前大了很多，但还是会根据所在地和兴趣特点等划分用户，以提供更细致的服务。

此外，我们团队的一个优势是和一些OEM硬件厂商配合，将《街旁》内置到手机上。现在合作的厂商有HTC、索爱、诺基亚、摩托罗拉等。这也带来了不少新用户。

《程序员》：LBS活跃用户比例偏低，如何提升用户黏度？

刘大卫：每周都会出现很多新的App，怎样提高用户体验提高活跃度，是所有App都应该考虑的问题，而不是LBS本身的问题。百度地图就是一个很好的LBS应用，它的活跃度就很高，因为它为用户提供的服务本身很有价值。

提高用户黏度的方法很多。《街旁》每次推出一个新版本的App都会出现一个小高潮，会引起很多人关注和使用，所以我们考虑加快产品的迭代速度。版本发布的时间间隔太长的话，会损失一些活跃用户。而如果一直有最新版本可用的话，就会实现新用户的自然增长。签到的意义在于可以及时分享，而记录是一种积累，意义会更大。《街旁》新的App会更专注于怎样帮用户做记录，将用户分享自己在哪里、在做什么的习惯变成真实生活的一种记录，吸引一些长期用户。

坚持走SNS路线

《程序员》：《街旁》的活跃用户是因为钟爱“签到”，还是深受徽章或折扣等服务的吸引？

刘大卫：我们在尽力为用户提供很有《街旁》风格的服务，无论是优惠、游戏还是徽章，都希望能为用户提供更“街旁”的体验。

LBS或O2O有很大的市场空间，大家都在做优惠或为用户带来惊喜，这时就要做出自己的特色和风格。我们的用户就很喜欢跟《街旁》风格的功能互动。虽然其本身的价值和其他同类产品差别不大，但形式很“街旁”，整体体验会好很多，用户增长就会比较快。这跟单纯登一些广告或给一些优惠有很大区别。

《程序员》：新浪微博、豆瓣等都提供了签到功能，《街旁》会不会有危机感？

刘大卫：它们从2012年就有了类似的服务。那时，我们确实很担心它们会和《街旁》重叠。然而，签到是一种很新的服务，用户群还没有成熟，小团队可能会更灵活，能很快捷地改变以适应需求上的变化。而对大公司或来说，这样做显然很难。

虽然几乎每个SNS都加了签到功能，但用户一般

只会选择一个签到应用来将信息同步到其他SNS上。而大公司的产品很难跨平台。例如，新浪微博不会和QQ同步。

还有一个应用本身的架构问题。新浪微博、豆瓣及其他一些大的SNS网络，都是基于PC架构建起来的，不管是信息架构还是内容策略都要在PC上和移动端平分。这时，《街旁》的专注成为了优势，只需将力量用在移动端，内容方面采用用户生成内容（UGC）策略。

《程序员》：《街旁》是打算走社交化路线？还是考虑依靠数据分析和处理做一些其他的商业模式？

刘大卫：走SNS路线，新浪微博和腾讯微信上的朋友圈会越来越大，而《街旁》则要提供能保持用户真实生活的朋友圈。这是《街旁》和其他SNS的互补，相信可以有很大的发展潜力。

《程序员》：签到服务收集到的信息能够反映用户偏好，与日常生活息息相关、非常有价值，有没有考虑利用一下？

刘大卫：今年已经开始做一些这方面的尝试了。用户的签到，很像PC用户访问网站时的Cookie。《街旁》现在服务很多品牌，它们越来越需要这方面的数据。比如，20~23岁的男性会经常光顾李宁或耐克等专卖店，但现在《街旁》的用户量还不够大，所以只能提供一些最基本的数据，不会特别深入。但长期看来，可能在1~2年后，会考虑数据方面深层次的使用。

LBS要勇敢前行

《程序员》：与Foursquare相比，LBS在中国的发展有何特点？

刘大卫：两者的差异有很多。Foursquare的本地商户从0涨到50万，是一个非常自然的成长，没有线下团队来经营。而在中国，与本地商户的合作需要有专门的团队来拓展和运营。相信Google和百度广告平台的情况也差不多。这是Foursquare和《街旁》的主要区别。

中国有新浪微博、人人和豆瓣等多家SNS。在这么大的SNS平台环境中，做一些与SNS有关的服务会比较容易。

《程序员》：LBS的变现能力遭到质疑，在提高盈利能力方面，你有哪些考虑？

刘大卫：提到LBS变现能力，通常都会想到如何与本地商户合作。在这方面，团购类网站做得很好，大众点评做得也不错。LBS想做类似的事，要先研究这两大平台的做法，做出差异化，或在它们的基础上做细分服务。可以考虑先提供一些免费服务，看看效果，不断地积累经验，形成自己的思路。

《街旁》的用户基本上是较大城市的中高端人群，消费能力比较高。目前，我们主要从品牌广告合作中盈利。与耐克、阿迪达斯、星巴克等一起做品牌推广。

《程序员》：你怎样看待LBS社交产品的生存现状？对此有哪些思考？

刘大卫：有一些会成功，但很多会失败。而只有失败到某种程度，才能看出一些比较成功的案例，所以现在下什么断言为时尚早。《街旁》也有一些短期的失败，其间我们看到了自己做错了什么，也看到了别的App在做什么、哪些对哪些错。经历这个过程之后，我们才有很大的改进，才可能在将来取得更好的成绩。

类似的例子是，七八年前互联网环境中很多门户网站失败，但正是经历了这个过程，大家才知道什么可以做，什么不能做。

移动互联网的现状，不管在美国，还在中国，很像当年互联网的情况，所以现阶段要很勇敢地去尝试。虽然风险很高、学习成本很高，但希望也很大。几年前，在互联网环境中有很多网站都失败了，但eBay却在极其困难的环境中发展成为一家大公司。在中国，一些大门户网站如网易、新浪和搜狐等也存活下来了。

因此，在早期环境中，团队要勇于尝试，多学习一些东西，随时改变以更好地适应发展环境和发展需求。

最值得关注的八家创业公司

点评人 / 王坚，金错刀

1 The Honest Company

创始人：著名女演员Jessica Alba和美国社交购物网站Shoedazzle联合创始人Brian Lee

创建时间：2012年1月

该公司主要销售无毒性的婴儿产品。Alba希望将The Honest Company打造成这种品牌，并以此提供安全且精致的用品，该公司将拥有自己完整的生产流程，从产品设计到生产再到销售都将包括其中。

点评：曾被Business Insider评为最酷的创业公司，想法很好：销售无毒或无化学毒性的婴儿产品。但这是一个巨大的红海市场，什么是Honest最独一无二的极致卖点？并不是很清晰。

2 Peek

创始人：前Artsy营销总监Ruzwana Bashir和前Pipewise工程总监Oskar Bruening

创建时间：2012年1月

该公司目前为止还不算是科技公司，但其刚刚宣布了新一轮的融资，以为即将推出的科技业务作准备。该公司将成为帮助旅行者查找和预订高质量旅游体验和旅游活动的场所。

点评：该公司的投资方包括Eric Schmidt，什么吸引了Google执行董事长Eric Schmidt的眼光？是人，还是项目？也许两者都有。如果高端旅游查找和体验能跟互联网有一个很好的结合，还是很值得期待的。

3 PandoDaily

创始人：美国科技博客网站TechCrunch前撰稿人Sarah Lacy

创建时间：2012年1月

该公司将是一家科技博客网站，即“记录硅谷的场所”。

点评：她要做TechCrunch2.0吗？答案是肯定的。PandoDaily的确继承了TechCrunch曾经的精华：1. 撰写高质量的科技内容和独家消息；2. 与著名科技写手合作；3. 创业发现。我相信PandoDaily会成为一个很不错的科技博客，但她到底能对TechCrunch的模式产生多大的升级，目前并没有看到太多动作。我认为，升级方向是社区化，但这可需要大投资，250万美元是不够花的。

4 Engagio

创始人：知名企业战略和全球化解决方案顾问William Mougayar

创建时间：2012年2月

Engagio是一个管理社交媒体评论意见和消息的“收集箱”，它不会捕捉每一条消息或Facebook的每个信息。相反，Engagio只是收集那些回复用户行动的信息。该公司主要从Twitter、Facebook、Google+、HackerNews、Tumblr、Foursquare、LinkedIn和Disqus等社交网站收集信息。

点评：社交网络优化是搜索引擎优化之后的一块大市场，因为社交网络正在成为最大的流量分配中心。Facebook对很多网站的引流已经超过了Google（中国因为QQ空间、微博都没花心思在内容生态上，所以百度还是中心），流量从哪里来，自然要优化哪里。这个市场大有可为，Engagio也确实在解决真实的问题。

5 CruiseWise

创始人：前Mamram产品经理AmitAharonir和Boston Coonsulting Group项目负责人Steve Davis
创建时间：2012年2月

该公司是一家为客户筹划航游行程提供搜索和预订工具的网站，同时也为游客提供旅游指导意见。
点评：据说2012年邮轮的市场规模将达到340亿美元，网络订票是一个很有价值的问题。从用户操作效率来讲，在一个地方搞定所有的旅游预订是最好的，Expedia收购CruiseWise，也许是美满的结局。

6 Banters

创始人：Lauren Leto
创建时间：2012年1月

该公司可以让用户使用类似Siri的技术往博客上发布文本内容。目前，如果用户将Banters的电话号码添加到地址簿中，那么用户所说的任何内容也都将可以张贴到该网站上。
点评：在互联网覆盖度如此之高的今天，以没有网络环境为出发点构建出来的业务真有点开倒车。一年半达到6万用户的情况下继续踩油门，祝福Banters在途中发现美丽的风景。

7 QuBit

创始人：两位前Google产品经理Emre Baran、Daniel Shellard和前Google营销经理Ian McCaig
创建时间：2010年1月

QuBit让内容所有者和电子商务网站可以获得消费者数据分析并提供更个性化的Web商务体验。让网站所有者不需要彻底修改其现有的内容管理系统或Web设计就能获得更多的收入。
点评：数据挖掘和数据分析，随着互联网的进化以及移动互联网的发展，绝对是一个黄金机会点。

8 Pinterest

创始人：Google产品顾问Ben Silbermann
创建时间：2010年3月正式上线

Pinterest创办于2011年，仅从字面上就能大致了解这家网站的定位：图钉（Pin）与兴趣（interest）。也就是说，它是基于视觉分享的社交网站，是“个人版猎酷工具”，用户可以把自己感兴趣的东西，用图钉钉在钉板上，或归类收藏，或与朋友分享。每个图钉背后也都藏着一个“故事”。
点评：Pinterest绝对是当之无愧的酷公司。目前，国内已经诞生超过30家类Pinterest网站，不过它的牛X你学不会：1. 史上没有一家公司比Pinterest更快达到美国单月1千万不重复到访人数。2. 2012年以来，每日用户成长已超过145%；3. 所产生的推荐连结流量超过YouTube、Google+与LinkedIn总合。4. 使用者男女比率为13 : 87。

本文部分资料源于Business Insider

中国最值得关注的IT从业者创业图



移动互联网

哈达在线（摇传）	李金波
PunchBox捕鱼达人	陈昊芝
Domob多盟	齐玉杰
同步网络	熊俊
N多网	陈驷
尔雅物云	张默/CEO 郭立峰/CTO
创新工场：	应用汇 豌豆荚 点心 磊友 友盟

云计算

酷盘	顾志诚
安全宝	马杰
行云	唐彬森
监控宝	郭欣
Ucloud	季昕华
Q盘	许式伟/CEO 徐立/联合创始人



电子商务

美丽说	徐易容	
优众网	陈啸/CEO	孙彤宇/联合创始人
翻翻网	边江	李顺顺/联合创始人

SNS

点点网	许朝军	
知乎	周源	
逛	朱宁（白鸦）	穆小斌/联合创始人

其他

钛金骑士	毛海滨	
i美股	方三文	
简悦	钟晖（叮当）/CEO	吴云洋（云风）/CTO

产品原则和产品评审团

文 / Marty Cagan 译 / 黄捷文，唐丰能

Marty Cagan是享有世界声誉的产品管理专家，曾经担任网景副总裁、eBay产品管理及设计高级副总裁。本文是他回顾自己二十多年来从事软件产品管理工作的总结和经验分享，描述了产品开发需要遵循的产品原则以及成立产品评审团的必要性。



Marty Cagan

过去20年，Marty Cagan作为负责定义和开发产品的高级经理人为多家一流企业工作过，包括惠普、网景通信、美国在线、eBay。他亲历了个人电脑、互联网、电子商务的起落沉浮，致力于通过写作、演讲、培训帮助客户打造富有创意的产品。为此，他撰写了《Inspired: How to Create Products Customers Love》一书，创办了硅谷产品集团公司（SVPG）。在此之前，他的最后一份工作是担任eBay产品管理及设计高级副总裁，负责规划全球电子商务网站的产品和服务。

产品原则

产品原则是对团队信仰和价值观的总结，用来指导产品团队作出正确的决策和取舍。它体现了产品团队的目标和愿景，是产品战略的重要组成部分。从形式上看，它是一系列明确的、体现团队特色的产品价值准则。

每次加入新团队，我要做的第一件事都是制定产品原则。这意味着要决定什么重要、什么不重要，哪些原则是根本的、战略性的，哪些是临时的、战术性的。

产品原则不是产品功能的清单，不依赖于任何单独的产品，而是整个产品线的战略指南和公司的价值宣言。好的产品原则还可以激发设计产品的灵感。

制定产品原则的过程也是学习的过程，可以从中了解新公司的企业文化，以及公司创始人设立的企业目标。产品原则是一套价值判断的框架，帮助公司作出正确的决策。

举例来说，某电影网站的产品原则是相信社区用户的影评比专业人士的影评更有价值。当某家制片厂想借网站发表评论时，产品团队就可以根据这条产品原则决定是否采纳。

产品原则是否公开因公司而异。它既可以用做团队内部的指导工具，如产品战略文档，也可以公开给客户、合作伙伴、投资人，用于向公众宣传公司的理念。

此外，产品原则还可以用来团结产品团队，让产品经理、产品设计师、开发团队和营销团队形成共同的价值观，在认识上保持一致。这是任何产品说明文档都做不到的。

注意，只罗列出产品原则还不够，还要按原则的重要性排序。所有产品都要既易于使用又安全可靠，但总有需要优先考虑的原则。最重要的究竟是易用性，还是可靠性？

制定产品原则时，容易出现以下两类错误。

- 原则过于空泛，失去了指导作用。
- 把设计原则误当成产品原则。比如，为用户提供清晰的导航路径（方便用户完成下一步操作）属于常见的设计原则，不是产品原则。

如果你所在的团队还没有制定清晰的、有关产品理念的产品原则，那么应该把大家召集起来，花点时间来讨论分析，确定团队最看重的价值理念。

解决意见冲突

不少产品经理向我抱怨，他们受够了没完没了的会议（既无议程也无结果），以及会议中的那些争论和冲突。公司高管还时不时打断会议进程，扔下没头没脑的意见，拂袖而去，留下他们丈二和尚摸不着头脑。

这种情况在产品决策过程中经常发生，原因主要有以下几点。

- 每位同事对产品都有自己的看法。
- 大家都非常在乎产品，明白公司盈利得靠用户，只有产品才能吸引用户。
- 许多人以为自己比其他人了解目标用户，事实上并非如此。

另外，产品团队大多不必向产品经理汇报工作，产品经理没有管理产品团队的实权。在需要产品团队的配合时，产品经理只能摆事实、讲道理，不能强制执行。所以产品经理总觉得施展不开拳脚，非常沮丧。

有时大家各持己见，僵持不下，只能请高管出面定夺。出现这种局面，说明沟通方式有问题。产品创意在辩论中可以得到完善，但前提是大家形成一致意见。请高管出面决策、解决冲突会激化团队内部矛盾，得不偿失。

制定产品决策的过程中存在的困难着实不少，且是不可避免的，因为建设性的辩论和论证是定义优秀产品的必由之路。不过即使认识到这一点，我也很难把争论当成一种享受。

为了鼓励创新，改善讨论效果，同时降低外界干扰，在作产品决策之前，应该先确定决策要解决什么问题，让大家在以下几个要点上达成共识。

- 究竟要解决什么问题？
- 要为哪类人物角色解决这个问题？
- 产品要达到什么目标？
- 每项目标的优先级是什么？

在我看来，每当团队内出现严重的意见分歧时，并非是大家对事实的认定有争议，而是对目标和目标的优先级有不同的理解。

比如说，团队首先应该确定哪个目标对用户最重要，是易用性、响应速度、功能、成本、安全性，还是用户隐私？只有先统一对产品目标和目标优先级的认识，大家才能在此共识上进一步讨论各种方案的合理性与可行性。

务必认真分析产品目标的优先级（从最重要到最不重要逐项排序），让团队达成共识。切不可把所有目标都贴上“关键”和“重要”的标签。一定要区分什么最重要，什么第二重要……

我常被请去解决产品决策中出现的争议，发现多数团队跳过了这关键的一步。由于缺少基本评估标准，每个人对目标和优先级的理解都不同，大家往往情绪激动，在细枝末节上争执不下。

即使大家已经达成共识，也应该在讨论开始前再次予以强调，最好把目标按优先级顺序写在白板上，这样每位同事都可以看到评估方案和制定决策的确切依据。

制定决策的过程和依据必须完全透明，不要让人觉得你只凭直觉判断。务必告诉大家决策的依据和理由，清楚地展示每一个决策环节。

激烈的会议争论会影响大伙的斗志和工作效率。如果再出现这种情况，请先回顾产品目标和目标优先级，确保大家达成共识。

制定更及时、更可靠的产品决策

即使对小公司来说，制定决策通常也是既耗时又费力的。产品公司需要一套机制让决策者和相关人员及时作出明智的产品决策。我认为成立产品评审团是最好的解决途径。

我通常不热衷于出席会议或参加各种委员会，但产品评审团除外。产品评审团让所有决策者坐到一起，为把产品推向市场共同制定决策，可以有效地加快研发产品的速度。

组织产品评审团的难点在于既要为高管制定产品决策、监督产品流程提供透明的信息，又要避免高管干预产品团队的具体工作，如参与产品设计。

不少公司都有类似的组织，但我认为最早提出这个概念的人是eBay前COO Maynard Webb。多年来，我一直在实践中不断规范产品评审团的具体职责，完善其流程。

产品评审团的工作目标

成立产品评审团的目的是决定产品战略方向，从宏观上监督公司产品的研发流程，合理配置资源。产品评审团不制定公司的商业战略，而是在给定商业战略的条件下，提出与之相匹配的产品战略。产品评审团的决策直接影响企业的运营。

产品评审团的成员组成

产品评审团由公司各个部门的管理者组成。虽然各个公司的情况不同，但通常都包括以下人员。

- 首席执行官/首席运营官/部门总经理
- 产品管理总监/副总监
- 用户体验设计总监/副总监
- 市场总监/副总监
- 开发总监/副总监
- 网站运营总监/副总监
- 客户服务总监/副总监

产品评审团的工作效果很大程度上取决于会议组织者的技巧。他必须不受干扰，善于阐明问题、促成决策。大公司里，组织者通常由公司的产品负责人担任；小公司里，则由老板担任。

产品评审团的组织者必须不受干扰，善于阐明问题、促成决策，才能确保评审团的工作效果。且要确保每个关键部门都有代表参加产品评审团，但人数控制在十人以内。

要确保每个关键部门都有代表参加产品评审团，但最好把人数控制在十人以内。如果有的部门不止一人参加评审团，应该选一个人代表部门陈述观点。例如，销售总裁代表市场部发言，质检（QA）主管代表开发部门发言。

产品评审团的职责

产品评审团并不是设计和开发产品的团队，它的职责是监督产品研发流程，制定关键决策。

它根据研发产品的四个里程碑来评审产品，制定决策。

1. 评审产品战略和产品路线图，启动评估产品机会的工作，即选择值得投入精力的产品，请产品经理开始评估产品机会。

2. 根据评估产品机会的结果，决定是否开始定义产品的解决方案。

3. 评审产品原型、用户测试结果、成本估算明细，决定是否开始开发产品。

4. 评审最终产品、产品品质、发布计划、社会效应，决定是否发布产品。

注意事项

- 小公司的产品评审团通常负责评审公司所有的产品；大公司可能需要根据业务单位的大小，设立若干个产品评审团。

- 产品评审团不负责评审对产品细节的更新或修正。这是为了加快对细节问题的处理，保证公司业务运作流畅。

- 产品评审团不负责具体的产品设计工作。如果产品存在缺陷，应该由产品团队着手处理，然后重新提交产品评审团评审。

- 在2号里程碑处，由于产品解决方案尚未形成，不可凭直觉估算产品的成本，至多只能估计大致的项目规模；但在3号里程碑处，应该仔细估算开发时间和成本，让公司上下做好准备。

- 尽量避免产品评审团讨论具体执行策略，讨论这些问题非常耗费时间。如果必须讨论，则一定要控制好时间，不要影响产品评审团监督产品流程的主要工作。

- 产品评审团开会频率取决于具体产品的进度，可以每月开一小时会议或者每周开两小时会议。

- 产品评审团还应该回顾、分析产品上市后的表现。可以在产品发布3~6个月后，请产品团队汇报产品的市场业绩表现，产品评审团可以反思之前的决策是否明智，今后应该如何调整。

- 每次评审会议，最好由产品经理向产品评审团汇报产品的进展情况。由产品经理的直接领导指导产品经理准备陈述内容，确保产品经理准备充分。在会议召开前，产品经理最好逐一向产品评审团成员做简要汇报，存在疑问应及时解决，避免在汇报过程中措手不及。

如果公司制定产品决策的效率太低，应考虑组织产品评审团。它可以替代以往各种冗长的决策会议，大大缩短决策时间，制定明智、及时、透明的产品决策。

何时估算项目成本

尽管软件行业早就有估算成本的传统，但直到今天仍然容易出现混乱的估算结果。我认为混乱的原因在于管理层总是希望尽早获悉成本信息，但开发人员往往要较晚才能精确估算成本（至少要等到具体的解决方案出炉）。结果，要么过早估算导致结果与实际出入很大，要么结果虽然准确，但远远超出预算，让管理层难以接受。

这里要介绍的估算方式虽然源自我提倡的产品研发流程，但同样能解决大多数公司的问题。

开始研发产品前，应该先评估产品机会。评估产品机会的目的是看产品创意宣称要解决的问题是否有价值。此时，解决方案尚未出炉，手头仅有产品创意和待解决的问题，所以产品团队只能先大致估算项目的规模（建议分成小型、中型、大型三个等级），再根据项目规模粗略估算项目成本。虽然这种估算与实际情况会有出入，但通常不会出现跨级别的误差。

确认产品机会有价值，粗略估算的成本也可以接受，管理层才能允许项目组着手定义产品解决方案。理想情况下，详细的产品解决方案还应该包含可供用户测试的高保真原型。

在定义解决方案的过程中，首先产品经理和交互设计师需要一位开发人员的协助来评估不同解决方案的成本。然后产品经理和交互设计师根据评估结果进一步调整解决方案。待完整的产品说明文档形成后，即可根据文档细节生成详细、可靠的成本估算结果。

此时，手头有了详细的产品说明文档、可信的成本估算结果，管理层可以很方便地决定是否开始开发产品。如果解决方案有问题或成本估算结果超出了预算，管理层可以马上叫停。如果决定开发产品，产品团队就可以在充分了解产品定义与成本细节的条件下全力开始工作。

总而言之，我建议分两个阶段进行成本估算。在评估产品机会时做粗略估算，根据最终的产品说明文档做详细估算。



本文节选自华中科技大学出版社《启示录：打造用户喜爱的产品》一书及作者的博客。该书从人员、流程、产品三个角度介绍了现代软件（互联网）产品管理的实践经验和理念。特此感谢华中科技大学出版社与Marty Cagan先生授权。

厂商资讯

SAP Business One中国峰会开幕

作为Business One在中国的首次峰会，此次大会于3月13日在北京拉开序幕，并陆续登陆上海、深圳、成都等城市。会上正式推出了可“按需随选”的SAP Business One OnDemand云计算解决方案，并通过现场演示，分享了基于内存计算技术HANA的商务智能应用。

2012英特尔软件大会瞄准MIC编程

本届大会于3月19日正式登陆北京，议题主要围绕基于英特尔MIC（集成众核）架构之上的并行开发编程和优化展开。作为高性能计算的未来，MIC架构在单个CPU芯片中融合了众多核心，通过使用标准的C、C++和FROTRAN源代码进行编程，大大方便了开发人员。而为MIC编写的源代码同样可以应用和运行在标准的至强处理器平台之上。

红帽进一步拓展企业虚拟化市场

近日，红帽宣布其企业虚拟化3.0正式上市，扩展了其在服务器和桌面虚拟化管理工具及基于内核的虚拟机（KVM）管理程序方面的能力。企业虚拟化3.0管理系统是在红帽企业Linux上的JBoss企业应用平台上运行的一种Java应用，可以提供超过1000项新特性、增强特性和改进项目。红帽企业虚拟化服务器版最初于2009年11月发布，并于2010年中期发布了企业虚拟化桌面版。

全面预算管理信息化研讨会召开

日前，由元年软件主办的“引领预算信息化新时代——实现高绩效的全面预算管理”研讨会在京召开。全面预算是企业发展战略的关键环节，费用管控是企业健康运作的有力保证，企业的全面预算与企业费用控制体系息息相关，只有通过好的信息化工具才能保障其正常运转。会上，中茶股份和劲酒集团分享了自己信息化工作的历程。

与未知同行
论敏捷开发中的反馈与反复

文 / 殷亮

我的思想应向我指示我正站在何处，但不该向我透露我将前往何方。我爱对未来的无知，不愿因焦急和对未来预先付出的代价而毁灭。

——弗里德里希•尼采（Friedrich Wilhelm Nietzsche）

理性王国的失意者

在古希腊的托勒密（Claudius Ptolemaeus）所构想的体系里，人类在宇宙内被指派到了一个中心位置，太阳围绕着地球朝升暮落、日复一日。一直到400多年前，哥白尼亲手将以天之骄子自居的人类从这一显赫的位置上驱逐下来。在1858年前，人类也一直认为自己是万物之灵，凌驾于自然规则之上，较其他生命高出一等。然而达尔文跑遍大半个世界后却告知我们：我们又错了，我们犯下了一种对宇宙的不虔诚罪。同样，在软件业诞生以来的半个多世纪的时间里，我们曾试图将编程与工业生产等同起来，高喊软件工程化与软件科学化，但时间再一次证明了我们的无知。

人类对于自己知道什么及未来会发生什么，总是过于乐观。因为人是这样一种动物——他们活在未来，行动在现在，靠着对未来的预测来决定当下的行动。关于未来，我们确有所知，但实际上也就“仅此而已”。然而却总有一种企图将一切置于理性王国之上的倾向，在灵魂深处挥之不去。也正是这种对不确定性的厌恶和对预测能力的盲目自信，使软件开发在很长一段时间内都只能黯然深陷于瀑布流程的炼狱之中。在Larry Constantine笔下的混沌年月里，“无数的管理者为保持控制力和稳定而努力，为软件性能和可预测性而竭尽心思……站在项目负责人的角度，他们试图理解和管理无比复杂的开发技术和流程，但技术变化的步伐却越来越快，从而导致复杂性不断增高，也更加难于管理”。先行者向着自然科学般的绝对精确前进，试图消除软件开发过程中的一切不确定性，但真理却放弃了他们。我们总是认为能知道更多就能做到更多，这种状态就像尼采所说的“精神越来越重，变成骆驼走进沙漠”，而精神的重负无疑就是通往敏捷开发路上最顽固的障碍。

沟通的极限

有一种决定论的思想——坚持认为世界的因果结构很强，如果给出世界在某一时刻的整个状态的完整描述，那么在这些因果律的帮助下，过去或未来的任意事情都能够被计算出来。当然，我们都知道，想以一己之力去了解“世界某一时刻的整个状态”，无疑是力所不逮的。但现在我们假设这是可能的事情，可以通过现在的描述，预测到未来的状态；那么，在软件开发中，假如我们知道客户现在想要的和团队当前的状态，是否就可以预测出将来能否顺利交付软件呢？Alistair Cockburn对此给出的回答是：“不可能”。Cockburn在《Agile Software Development: The Cooperative Game》一书中详尽阐述了沟通的不完全性，并认为完全的沟通永远不可能实现。换

句话说，即使客户知道自己想要什么，但也很难用某种语言或文字将信息完整传递给我们。这样，我们都不需要去争论是否可以通过现在预测未来了，因为在真实世界里，我们甚至连准确理解现在身边的一切都做不到。

这种沟通的不完全性，是由于语言和文字中的歧义与含糊不清是不可避免的。我们日常生活中的种种误解及对误解的解释，是每一个人的人生中都共有的经历。当然，这并不是说语言与文字的歧义是无法消除的，像我们所归纳的设计模式和所创造的UML语言，实际上都是在消除歧义方面的一种努力。通过设计模式和UML，我们在沟通方面的精确度有所提高，但随之而来的却是明晰度和可读性方面的损失。另外，学习曲线陡增与应用范围狭窄，却使得这种方式很难推广到我们生活中的每一个领域。因此，我们很可能永远也无法创造出一种精确的需求语言，让客户去学习并准确描述其需求。就像我们永远也不可能在软件开发过程中让团队放弃一切口头与文字上的沟通而只使用UML一样。

反馈：他山之石

在一切需要创造全新事物的活动中，都有一定的对未来的不尽知，这种不尽知在其他领域里也如此，而不独软件开发为然。画家在开始作画前脑中所构想的，与最终完成的作品会有所不同；电视剧，从最开始的剧本编写到开播，剧情也必然存在出入；而管理者，如果只能在给定清晰的目标以及完全的信息后才能行动，那么必然不能胜任岗位，在这个变化的世界中处理问题。从这些相似的领域中，软件开发者能学习到什么呢？

很显然，画家无法在脑中构建出一幅完整的图像后再开始动笔，因为大脑对于成像而言快速有余但细节不足，难以保存且缺乏稳定性。画家所能做的，不过是参照着脑中的模糊影像，有了一定的准备后便开始下笔。之后，他每画完一笔，画布上都产生了一个新的版本，渐渐地，他下一笔的位置不再依赖脑中的影像，而是更多地根据眼前的中间版本来调整。脑中的影像映射了整个轮廓，但并未暗示其中的任何一笔。

而就电视剧而言，由于其中涉及大量情节和角色，所以在拍摄之前将剧本冻结无疑很难。因此，在拍摄过程中根据现场的展现来不断完善剧本是必不可少的过程。在美国和日本，电视剧甚至都不是一次性制作完成的，而更多采用边播边拍的模式。制作公司根据观众的反馈来调整接下来的剧情发展，使之更符合他们的喜好。

通常来说，企业决策所针对的“事件”都是独一无二的：一方面是由于不可能有大量相同的或类似的其他事件，足以让我们获得近似的概率；另一方面，企业决策通常都依赖于对未来的展望与对人的展望，而时间与人的不可控使得每个决策都不能等而视之。1924年，在Alfred Sloan, Jr.的领导下，通用汽车公司开始实施一种全新的生产计划控制政策：要求各汽车分部每十天从经销商处获取实际销售信息，并与之前的月度预测相比较。然后每月再对整个形势进行一次仔细的分析以确订当初的预计是过高还是过低。如果认为预测过高，生产计划量马上削减；如果发现零售需求大于原来的估计，生产计划量将增加。“一言以蔽之，公司并不试图事先制定一个僵硬的年度计划，然后不论零售需求如何变动均将其坚持到底，现在公司悉行的政策是将生产始终置于控制之中”【摘自1926年通用汽车管理人员向美国管理协会的介绍】。而在此之前，整个汽车产业都是以死板的年度预测来做有关产量的决策，这种忽视未来不确定性的方法实际上是导致1920年与1924年整个汽车产业溃败的主要原因。

在各个领域里，人们在创造全新事物时，所运用的手段有一个共有的特质：迅速执行，然后频繁调整。因为要在一开始就知道未来的一切，是惊人的奢侈，可能是我们都负担不起的。而如果将整个创造过程看做一个封闭系统，直到最后一刻才打开的话，那么一旦与现实有偏差，返工成本又可能是我们所难以承受之重。因而我们强调创造过程中的反馈：就像一个射击手，从开始瞄准到扣动板机，一直都在依据自己的所见来调整角度，而最终角度通常很难在开始时就能确定下来。同样的，画家通过眼前所见得到反馈，电视剧制作公司通过观众得到反馈，而管理者则通过真实的执行情况得到反馈。倘若他们不这样做，

就只能通过自己的大脑来反馈，但从大脑计算中得来的反馈怎会比双眼所见更清晰、比观众评价更有效、比事实证明更真实呢？大脑有它的伟大之处，但大脑也有受限的一面。

因此，在敏捷价值观里，我们强调客户协作，也就是承认从客户处持续得到反馈和从实际的构建活动中得到反馈的重要性。客户也许一开始并不知道他真正想要的是什么，但他们可以从我们交付的中间产品中得到启发；我们也是一样，也许一开始并不能完全理解从客户那里得到的需求，但除了客户，我们还能从谁那里得到这些呢？除了与客户协作，我们还能通过哪种途径去接近双方共同的目标呢？我们别无选择。同时我们还看到，无论是在开发过程中引入对模糊性的容忍，还是强调反馈的重要性，实际上都不是软件业的新发明，我们不过是在照搬其他领域的最佳实践而已。然则他山之石，可以攻玉。

反复

前文讨论到，在一切需要创造全新事物的领域里，都存在对未来的不确定性。之后还找到了应对这种不确定性的有效方式——反复进行，而不是一步到位。但通过对现实世界的观察，我们也不难发现：虽然同样是创造新事物、同样存在不确定性，但反复方式并不能适应于所有领域。比如建筑业，很少见到不断“推倒再部分重建”的施工方式；而在医疗业，虽然在某些特殊场合，医生也会通过验药物、观察患者反应来渐进治疗，但大多数情况下，这种方式显然不可取。那这些领域与前文讨论的情况有什么不同之处呢？

先看一下建筑业，在施工过程中进行结构调整无疑会大费周折，再小的变更也可能导致大面积重建，原料与工期的损失都将十分可观；医疗业也有类似建筑工程的困境，医生面对的是无价的健康，任何未经深思熟虑的决定都可能给患者造成不必要的伤害，这一成本无疑也是不可估量的。然而软件开发则不同，我们可以通过修改有问题的代码行调整未满足需求的模块，然后重新编译，这一变更的成本往往极其廉价——而这种廉价甚至给行外人士一种错觉，认为在已有的基础上增加功能或变更功能几乎不需要额外成本。

由此可以推断出，反复进行的一个重要条件是可以廉价变更。除软件开发外，这一点对于艺术构建与剧本编写也是显而见的，且这两个领域不仅可以廉价变更，似乎还有种可称为“将错就错”的优势。在一个信息顺畅的企业里，决策变更通常也能迅速得到执行，尤其是近年来随着实物期权（real option）理论的发展，企业管理者决策时往往会着重考虑后续的灵活性。反之，在变更成本非常高的领域，如乐团演奏、航天工程等，都需要事前充分准备，因为一旦开始则难以反复。

由于在一些领域里，存在着廉价的变更方式，使得“反馈—变更”成为一种可能的战术。因此我们也可以说，对于反复方式而言，存在中途变更的可能性且代价不高，是它的必要条件。但我们还不能将其视为充分条件，因为在一些领域中，虽然变更并不会导致昂贵的开销，但人们并不采取反复的方式来进行。比如一般的租赁协议，尽管在双方都同意的情况下，变更合同的成本是廉价的，然而先短期试租再根据反馈签订长期合同的情况并不常见。另一个例子是农场种植。假如某一农场想种植新的作物品种，通常也不会先通过少量种植获得反馈后，再进行大量种植。这是因为，在这些领域里，虽然未来也存在一定程度的不确定性，但这种不确定性的波动范围很小。上面我们提到的合同约束下的租赁、某一地区的气候，这些实际上都具有相当程度的稳定性，所以即使有廉价的变更方式，人们也不会以反复的方式去安排他们的计划。我们可以将上述讨论总结如图1所示的模型。

在这个模型中，我们提取了决定反复方式价值的两个维度：**响应能力**，在实施过程中变更的可能性和成本；**不确定性**，未来变化的不可知程度。在这两个维度上的取值都很高时，以反复方式进行的价值最高，软件开发便是最佳代表，因此敏捷交付在这一活动中具有极其重要的战略意义。另一个极端表现为低不确定性与低响应能力，在这里以建筑业为代表：建筑业的低响应能力，在前文已讨论过，它降低了建筑工程的灵活性，但这只是其中一方面；另一方面，由于有数千年经验的积累，建筑学和物理学科已发展得非常完

善，其中的不确定性大大降低，因此，在建筑业中反复方式几乎没有任何价值。而其他两个象限，一个受低不确定性的影响，另一个受低响应能力的影响，使反复方式的价值大打折扣。因此虽然可以使用反复方式，但并不具有明显优势。现实表现是：在特定情况下，可以观察到采取反复方式的样本，但并不普遍。

当然，也并非所有软件项目都具备高不确定性与高响应能力，如我们所观察到的，某些内部系统或模仿型软件产品，项目的不确定性并不高，通过反复的方式来调整的意义不大。也就是说，我们无法从根本上同意这种极端的说法，认为敏捷方法对一切软件项目都是最优选择。这也是很多敏捷图书常常误导读者的地方。实际上只要某种理论——它陈述的是一种可被观察到的事实规律，那么它就必然有它的前提或约束条件。正如哲学家Rudolf Carnap所指出的：“有时会认为某个规律是基本的，但后来却证明出其依赖于一定的时间地点或一定的事件……在许多领域，如生物学、社会学、人类学、经济学，有些规律初看起来好像普遍成立，但这种情况只是由于作者没有越出他的国家、大陆或他的历史时期的局限看问题所致”【Rudolf Carnap，1966，《An Introduction to the Philosophy of Science》。Carnap是逻辑经验主义创始人之一】。因此我认为，对于每个敏捷专家来说，他们的任务都不应该是急切地去证明敏捷方法有多么广泛的适应性，也不应该在自己的著作里加上“经我试验证明，在规模上百人的团队中敏捷仍然有效”的文字，这些都不是科学的精神。我们只有理解敏捷背后的机制并认清当前项目的特点后，才能确定是否应该以反复的方式来推进项目。一切理论或工具都有一个共同的缺点，那就是它并不能防止人们对其滥用。

量子力学的启示

1927年，后来的诺贝尔物理学奖得主Werner Heisenberg首次提出了“不确定性原理”（Heisenberg uncertainty principle）。该原理指出，在一个量子力学系统中，一个粒子的位置和



图1 反馈-变更的必要条件

它的动量不可被同时确定。这种不确定性也是量子力学中基本规律的重要部分。更重要的一点是，它意味深长地告诉我们现实世界是“非决定论”的。在这个由不确定性主宰的世界里，唯一能确定的就是我们对未来的不尽知，而这种不尽知，也是整个敏捷理论的核心假设。在2005年的敏捷开发大会上，对不确定性的坦诚更是写入了《相互依赖声明》（DOI）：“我们（预料到了不确定性）通过迭代、预测和自适应来管理不确定性。”注意，是“管理”而不是“消除”不确定性。它们的不同之处在于：前者是我们对不确定性的友好接纳，而后者则是在这个由理性支配的世界里光亮无比但却永难实现的梦想。

我们现在拥有的与软件开发知识，比以往任何时期的软件工作者都要多很多。但同样的，我们的知识也肯定要少于100年以后的软件工作者。知识与规律在历史长河中不断地被修正，我们无法得知这是一个无限过程，还是最终会走向某种尽头。但在所能预期到的未来里，如何管理不确定性，仍然会是学科中最重要的课题；反馈与反复的组合，也仍然会是组织开发过程时最应考虑的方式。而我们必须继续与未知同行，并容许我们今天否认同样曾属于我们的昨天。



殷亮

创新科存储技术有限公司项目经理。目前在深圳工作，研究领域以产业竞争与组织理论为主，6年软件项目管理与组织顾问经验。

责任编辑：杨爽（yangshuang@csdn.net）

应用程序设计语言的复杂性

文 / 周爱民

可以用四个关键词来区分开发规模的不同等级，分别是功能（Function）、程序（Program）、应用（Application）和系统（System）。其中，Application这个等级的开发活动被称为应用程序设计。本文是对这一规模下的语言的反思。

无论如何，工程师最终都要通过一个产品实现过程来满足用户的如下需求。

- 非功能性需求。
- 非当前需求。

这些是应用系统复杂性的主要构成。通常，工程师会将它们映射为系统中的两个主要概念，即“约束”与“变更”，如图1所示。

工程师对实施这些概念的具体技术和方法并不陌生。例如，在早期软件开发的实践中，非功能性中的跨平台问题，就通过伪编译指令，即编译期的约束来指定目标程序的运行平台，又通过命令行参数，即运行期的约束来指定目标程序的特定配置。目前来说，这些实现可以归纳为这样的一些思想与技术，如图2所示。

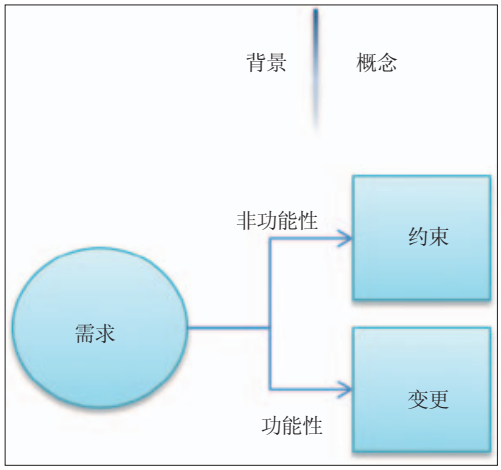


图1　约束与变更

运行环境与应用容器

应用程序并不直接运行在机器环境中，它通常依赖操作系统，甚至依赖操作系统的具体版本。严格说来，所谓“操作系统或其具体版本”提供的，就是一个运行环境。基本的操作系统主要封装了对硬件系统的访问，这包括：

- 存储（内存、硬盘和移动存储管理）；
- 标准的I/O设备（显示器、键盘、鼠标以及打印机等）；
- CPU及其分配（进程与线程管理等）；
- 网络、媒体、通信等。

除此之外，操作系统提供文件以及其他操作系统资源的访问。但并不是所有的操作系统都提供文件访问支持，因为这取决于该系统对“操作系统资源”的定义。例如，Windows系统中将注册表理解为资源，而Linux系统中没有注册表。

总的来说，操作系统为上述内容提供的API可以理解为运行环境的界面。但多数应用程序不只依赖这个环境，例如使用Visual C++编译的应用程序还依赖微软发布的VC++运行库。对于微软产品来说，操作系统中通常会带有该运行库的一个或多个版本，但某些情况下，仍然需要应用程序自己来管理这个库的安装与维护。

因而应用程序语言中的许多编译指令、伪指令、条件编译指令，甚至配置信息文件等用来做环境设定和依赖分析，简单点的有汇编语言中的

“.386”伪指令，复杂的则类似于Linux系统中的Make文件。而Make文件也有很多种类，并不只出现在Linux系统或某些面向Linux系统的语言中。事实上，几乎所有稍具规模的应用程序语言与开发环境都有自己的Make程序和文本。为了提供更为复杂的Make功能，许多Make文本被设计为“另一种语言”，也带有逻辑、条件、变量等语言特性。

但这些特性并不是相应“应用程序开发语言”的内容，而仅仅是用来将这个程序的产生、分发等过程中的一些需求加以规则化、程序化，因此我们通常也称之为Make脚本。

通过对规则的抽取、细化，以及提供脚本化的支持，“越来越复杂的Make”成为一个发展方向，几乎所有在“运行之前”能预料到的需求都可以通过以下几种技术来解决。

- 再编译一次。
- 分发不同的编译版本。
- 提供独特的安装过程。
- 特定的运行期环境依赖声明。
- 调整程序的启动参数。

然而，这种方案有如下三个致命的弱点。

- 其一，我们不可能预料到问题的全集。
- 其二，我们为可能出现的问题所付出的代价，甚至多于编写应用程序本身。
- 其三，由于这一方案（从逻辑上来讲）容许无度的需求，可能因此特性过于丰富而导致最终产品失控。

这些弱点的一种表现是：Linux下丰富的产品生态（如不同的编译、发布与维护版本），以及非功能性需求的实现上过于复杂（如安装）。

与此相对的，应用容器通过将应用限定在一个明确设定的环境中，有效地规避了上述问题。正因为环境特性由容器来决定，所以应用程序的开发与使用都不会超出容器提供的特性集的范围。

应用容器通常明确设定了应用的类型，例如Web应用容器或企业应用容器等。容器会根据应用的需求来搭载相应的可选件，并通过统一且一致的接口提供给应用程序使用。这些选件与接口被标准化，变成应用容器整体方案的一部分。

更进一步说，在这些（范围确定、接口标准）方案周边，部署方案、优化方案、硬件配置方案、商业应用方案等渐渐得以形成。

对应用与应用容器进行标准化，勾画出了一个类似于“集装箱”（参见《The Box》一书）的王国，EJB（Enterprise JavaBeans）是这个方案中的成功范例之一。

考察EJB的特性及通用应用容器下的规范，其本质上仍然是在“规则化”这一方向上，通过以下两种技术来实现“将非功能性需求限制在特定范

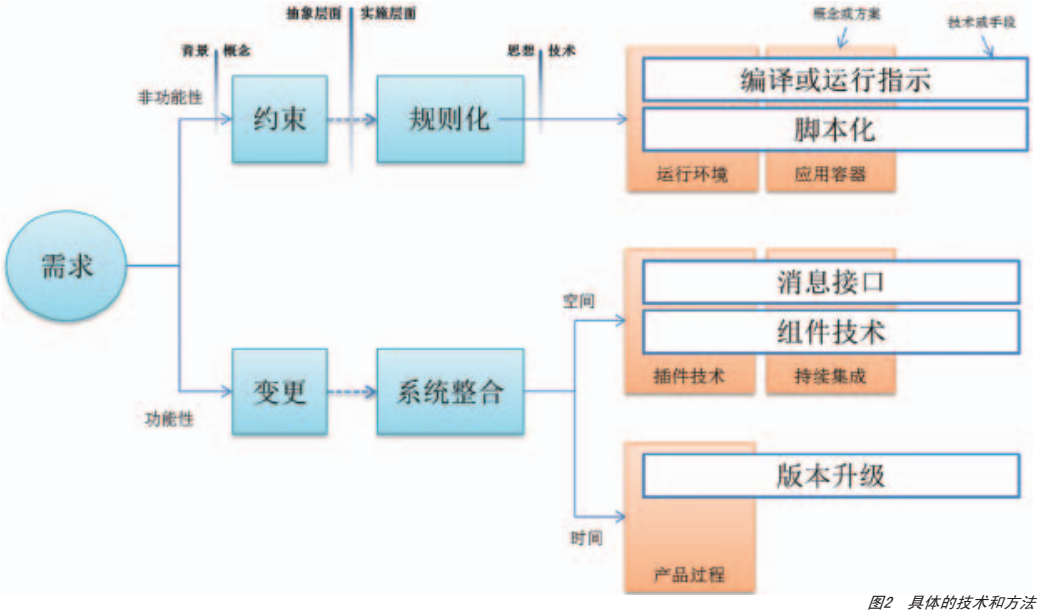


图2 具体的技术和方法

围”的。

■ 编译或运行指示，如注解（Annotation）、验证器（Validator）、配置文件。

■ 脚本化，如事务脚本（Transaction Script）。

插件与持续集成

对于一个应用程序来说，“插件”往往是一个动态链接库或静态链接库，或者一个独立发布的脚本包。如前所述的库或包，只是一堆代码在组织上的措辞而已。因此插件的本质在于用组织形式来解决空间因素所致的复杂性。

插件的宿主通常可以脱离部分或全部插件独立运行，而应用程序通常不能独立于模块/单元来完成编译。判断能否代之以插件技术的可行方法，就是判断能否从“模块/单元”中去除强约束的依赖性。

“插件”这一方案意味着插件的宿主（HOST）与插件本身（Plugins、Addons或Extensions等）是分别发布的。事实上与应用容器有一定的相似性，只是二者应付的问题集存有稍许差异而已，如下所述。

■ 应用容器主要提供可配置的、可重新实现的运行环境，并通过标准化来提供跨业务领域的应用支持——尽管应用容器也有应用范围的限制，但通常该范围的边界更大。

■ 插件的宿主通常由产品、用户和功能确定，技术方案也通常是预先可确定的，因此“宿主+插件”更适宜作为一个应用产品的具体解决方案。

与应用容器类似，宿主也有跨平台的问题，例如火狐浏览器。这个问题有两种解决方案。

■ 如果产品的用户有能力在不同平台上编译并重新发布宿主，或者宿主是针对特定平台开发而无须重新发布的，那么插件通常实现为与环境相关的二进制模块。例如Apache采用“WWW服务器+扩展库（动态链接库形式）”的形式来交付，一方面是考虑到二进制模块的执行效率，另一方面也因为编译发布是部署人员应有的技能。通常来说，Linux用户会经常面临这类情况。

■ 如果宿主有明确的跨平台需求，而其编译过程复杂或不宜公开编译方案与代码，或者用户没有能力和必要去区别、编译不同平台中的插件版本，那么插件通常实现为文本形式的脚本或预先编译为跨平台的中间代码/代码包。前者，如火狐浏览器中的扩展名为.xpi的插件；后者，如Java的.class文件。

插件技术是基于应用开发技术的“模块/单元”的理念而来的，不同的是，宿主通常可以脱离部分或全部插件独立运行，而应用程序通常不能独立于模块/单元来完成编译。判断能否代之以插件技术的可行方法，就是判断能否从“模块/单元”中去除强约束的依赖性。

插件提供了通过剪裁功能来重新定义产品的不同版本（如标准版、专业版等）的能力。与此同时，正因为它在概念上继承自“模块/单元”，所以也是一个“化整为零”的典型方案，使得在大型项目中实施持续集成有了一定的可能性。

其中的一个有趣的事实是：如果集成需要所有“模块/单元”的参与，那么集成的失败率就会因系统规模达到一定程度后出现级数性的增长；只有被集成的产品可以通过类似插件的机制，将单一部件的规模控制在一定范围内之后，持续集成（而不至于阻碍整个团队的推进）才成为可能。

复用中的问题

在面向对象设计理念中，“类”替代了“function/unit/module…”等组织方式，而在面向对象设计中，“HOST类+插件基类”是这一解决方案的基本设计。更进一步说，由于名字空间用于应付更大规模的组织，并且通常也具有“作用域”限制的能力，所以使用“HOST框架+（插件的）名字空间”也是一种可行的实现。

本质上，它与前一种设计并无不同，只是应用规模更大，而且作用域范围控制更灵活，类似安全性、账户等级限制等特性也往往更加容易实现。

但基于面向对象的插件技术的问题在于：通常

宿主和插件都基于相同的“面向对象语言”，因此具体的对象实现技术、二进制编译技术、对象封装技术等限制了插件的通用性。例如很难用Delphi开发一个“类”，并将它作为一个插件用在VC开发的宿主上。

这个问题的核心在于如何实现“组件复用”。这里的“组件”与此前的“模块”的区别在于：组件通常用于表达面向对象中的可复用对象/类，而模块通常用于表达结构化编程中的一个可复用件——后者并不一定是源代码中的一个单元（Unit），而是指在指定的应用开发环境中，由语言或平台决定的“可复用单位”。

这二者所面临的问题是完全一致的，其解决方案也有着延伸关系，例如Windows中的COM复用与DLL复用其实是一脉相承的，而.NET中的Assembely复用与COM复用也是类似的关系。

这通常涉及三种不同类型的复用（事实上划分它们的依据也并不唯一）：

- 二进制复用，如COM；
- 公共指令集复用，如Java类；
- 文本复用，如脚本。

无论技术方案提供商如何渲染这些技术，其最终结果都是一样的：开发人员终于可以用Delphi写一个COM组件，并让它运行在一个VC写的类工厂中；或者用任意的Java编译器编译一个.class文件，并分发给其他编译、运行环境下的宿主。当这一切成为现实（我的意思是组件复用成为应用开发环境的一个内置技术，例如Java beans）之后，组件技术的核心便集中在了宿主框架的设计、插件的加载机制，以及隔离宿主与插件之间或多个插件之间的安全性这几方面。

尽管并不明显，但事实上这样的隔离通常与消息机制有关。无论是从“消息”这一概念的历史，还是它在应用系统中的使用来说，都是“体系性”相当明显的一种技术。

语言的演进

现在，大多数程序员都可以写出一个具有典范意义的“Hello World”程序：

```
class HelloWorld {
 public static void main (String args[]) {
  for (;;) {
    System.out.print("Hello World ");
   }
  }
}
```

同时也会真正地从个人意识中忘掉这样一个程序的含义以及需求不过是：

```
print("Hello World ");
```

如果仅以这段程序而言，用户需求仅用上述代码即可实现，而Java（或其他一些应用程序开发语言）在代码中加入了以下概念。

- 类与对象等，如class HellWorld。
- 命名空间、导入导出等，如System.out以及隐式地导入System。
- 方法、属性、事件等，如print。
- 类方法、类静态成员等，如static。
- 引用、值与无值，及基本类型系统等，如void。
- 可见性、作用域等，如public。
- 字符串值、字符串类等，如"Hello World"与String args[]。
- 应用入口与运行环境约束等，如void main()。

这些概念大部分与“面向对象”这一语言范式有关，部分则出于应用环境、语法习惯等在语言的具体设计上的选择。

除了这些概念，在具体的开发环境中还会有容器、包、配置脚本、服务、模型、验证器、指示字、伪指令、分发、部署、版本容器、基线等概念，以应付不同角色的需求。

当一门语言从“实现程序功能”变成要“实现产品需求”时，其内部的语言设计思想，渐渐开始不遵守“算法+数据结构=程序”这一经典法则。整理这些在语言变化中的细节，可将这一切变化及可预期的、语言进化的方向都归结为：**通过在程序组织上的结构化来解决规模问题。**



周爱民

资深软件工程师、架构师。有十余年软件开发、项目管理、团队建设的经验，曾任盛大网络平台架构师、支付宝业务架构师等职。著有《JavaScript语言精髓与编程实践》等书。

责任编辑：杨爽（yangshuang@csdn.net）

创业公司如何实施股权激励

由于启动资金有限，很多创业公司都想通过实施股权激励方案，来吸引人才、稳定团队。但如何实施才能更有效呢？且听本期三位嘉宾将他们的实践经验娓娓道来。



宋崇宇
北京市中银律师事务所合伙人律师

股权激励是当然之选

如何增强员工的向心力？如何稳定员工？这些问题是困扰广大创业公司的难点和重点。而考虑到创业公司很难通过提高工资和福利的方式来实现员工队伍的稳定，股权激励便是当然之选。

全面考虑双方情况

创业公司应该分析员工和公司两方面的情况，来确定较佳的股权激励方案。

从公司的角度来说，主要考虑的问题是：如何用较低的现金成本稳定员工、如何锁定和促进员工的长期贡献。从员工的角度来说，主要考虑的问题是：如何从公司的业绩增长中尽快获得收益、如何从公司未来的高成长中获得未来的利益。

解析实施方式

创业公司实施股权激励的方式有如下几种。

- **虚拟股权**。员工获得虚拟的股权比例，并不获得实际的股权，获益与对应公司股权的分红相关。
- **股权增值权**。员工按照一定的股权比例获得对应的股权增值收益。
- **股权直接授予**。在一定的条件下，直接向员工授予公司的股权。
- **股权期权**。授予员工在未来某一时间购买或获得股权的权利。

如果创业公司在短期内业绩增长有限，那么虚拟股权和股权增值权的方式均会给公司造成较大现金支付压力。而且虚拟股权的方式并不会对员工的长期表现产生较大的激励作用，因为员工的收益与分红有关，就算业绩下滑，也可能会得到分红。尽管股权增值权方式能将员工利益与公司未来业绩的增长挂钩，但员工未得到实际股权，并不和其他股东一起承担公司经营风险，因而获得的激励作用也有限。

采用股权直接授予方式，员工获得的是实实在在的“股权”（一般情况下授予人是公司的大股东），公司可以用这种方式代替现金支付，从而节省成本。另外，员工获得股权后，其利益便与公司整体利益挂钩，这对锁定员工的长期贡献具有重大意义。从员工的角度来说，一旦获得真实的股权，则不仅可在公司有盈利时获得分红，而且可以股东的身份分享公司未来发展的收益。

就股权期权而言，可以通过设定获得股权的对价从而将员工的收益与股权的增值锁定，

因此在激励效果方面比股权直接授予要强许多。但获得股权的对价、获益的评估等因素对股权期权的实施非常重要，这便涉及公司价值评估的问题。对于IT行业的创业公司来说，公司的发展可能非线性，也可能根本无规律可循，因此公司现在价值的评估及未来价值的预估都是比较困难的问题，而且容易导致员工和公司之间出现分歧。在这种情况下，IT行业的创业公司如果实施股权期权存在一定的不确定因素。

综上，对于IT行业的创业公司来说，建议先对员工实施一定数量的股权直接授予以实现员工的基本稳定，然后，设定业绩增长等指标或明确工作时间等条件，分期分批实施股权期权。最后要强调的是，股权激励的工具是“股权”，其对于公司来说是一种“稀缺资源”，因此，需要严格界定被授予者的范围，同时还应谨慎设计方案，争取用最少的成本达成最大的激励效果。



郑友才
摩安CEO

留住人才，稳定团队

在移动互联网领域，真正的资产是人才，如何留住人才是创业成败的关键所在。本文将主要讨论创业公司如何通过实施股权激励，来留住人才、稳定团队。

创业型公司各方面的资源都非常有限，所以在分配和利用资源时一定要精打细算。加强新人的培养，不仅可为日后进行人才梯队管理做准备，还可以节约大量的薪资成本。然而，新人会逐渐成长为行业老人，面对公司很难开出诱人的薪资的现状，难免会心理失衡。因此，创业公司需要强化公司奋斗目标，增强员工对公司的归属感和认同感，同时应采取相应的物质激励措施。从长远发展来看，股权激励是很不错的办法。

实施股权激励一般有股份和期权两种方式。无论采用哪种方式，都要有相关条件，如员工必须在公司工作连续两年、离职后期权失效，等等。目的是让员工与公司一起成长，一起奋斗，一起分享。

创业公司可以拿出10%～30%的股份，奖励优秀员工，留住人才。有两种操作方式：一是直接无偿给予员工，通常员工占股比例较少；另一种方式是有偿给予，要求员工出部分资金来购买公司股份。出资比例因各个公司的情况而定，常见比例是1:1，即员工购买1股，公司给予1股。

期权方式是大多数互联网公司经常采用的。在公司上市或并购后，员工能从中获取相应的报酬。中国法律上没有明确承认期权这种方式，属于公司与员工之间签订的一种民事协议。但在西方国家是予以认可的。通常的做法是拿出公司股份的5%～15%作为期权池，划分成若干股期权，对每股期权单独定价，根据员工的贡献和职位高低配以不同数量的期权。操作起来相对简单、有效。

股权激励是一种长期激励措施，是保持员工积极性、创造性的有效手段。但人都是有惰性的，创业公司需要不断刺激员工，让员工不仅能看到长远的发展，也能追求到短期利益。短期激励主要是奖金、福利手段。收入稳定的创业公司，可以按每个季度对员工进行奖励，最有效的方式是按月对员工予以奖励。福利手段，就比较多了，比如来自外部的培训和交流，及公司组织的各类活动、商业补充保险等都是让员工认同公司的一种有效的短期激励手段。



肖鹏
阿普创新创始人兼产品总监

巧施股权激励，招贤纳士

对互联网领域的创业公司来说，人才尤为重要。能否吸引优秀的人才加盟，是决定公司未来能否成功的关键。“良禽择木而栖，贤臣择主而侍”，要想吸引优秀的人才，必然要提供足够的发展空间和比较丰厚的待遇。公司初创阶段，包括资金在内的各方面资源都比较紧张，没有能力单纯用高报酬来维持激励。这时，股权激励就成为创业公司招募人才并激励他们努力工作的重要手段之一。

合理的薪酬是根本

股权激励只是一种激励手段。按照马斯洛的需求层次理论，股权激励更偏向于被尊重和自我实现的需求，属于高级需求。而合理的薪酬制度属于生理和安全的需求，要优先满足。首先是基本薪酬制度设计必须合理，然后是以浮动薪酬调动核心人员积极性，最后是股权性薪酬激发管理层及核心人员与公司共同发展的积极性。而很多创业企业一上来就讲股权激励，在没有建立起完善的基本薪酬制度的情况下，搞股权激励显然是本末倒置的。

需注意的四个问题

创业公司在实施股权激励时，最容易出现问题的地方就是创始人和公司员工对公司未来估计的不一致。尚未经历过融资的创业公司的股票价格还没有进行过公开公平的估值，因此股票价格的确定具有主观性和不透明性。由于创始人对公司倾注了极大的 心血，很容易形成错觉高估公司的价值；而被激励对象往往因为创业公司条件简陋、各方面福利待遇和手头可供调用的资源不能与大公司相提并论，从而低估公司的价值，认为获得的股份或期权是鸡肋，不足以补偿自己的付出。所以，与公司员工充分交流，对公司的价值有比较一致的预期是非常重要的。

其次，空头支票也要尽量避免。在实施股权激励时，有些创业公司往往把前景描绘得特别美好，但基本实现不了。因为员工不是傻子，这样的空头支票没有任何作用。即便员工一时相信了，但一旦意识到公司的真实情况，就会感到受欺骗，产生反效果。

还可能出现一种情况。在公司真正达到股权激励条件后，创始人却突然反悔，觉得当初定的条件太宽松而不愿兑现或打折扣，这样也必然影响士气。

最后，要注意免费午餐造成的不良后果。拿到股份很容易，很可能反而导致员工失去工作动力，起到反作用。

小规模公司，全员持股

在规模较小的互联网创业公司里，每个员工都很重要，所以比较适合实行全员持股计划，由实际出资创始人持实股，核心人员持干股，其他员工给以期权。至于拿出多少股份来做激励，要依据公司的价值、员工的能力和贡献等，没有一定之规，分寸需要创始人自行把握。同时，也要参照激励员工的个人意愿。如果员工更愿意拿高工资、低股份，在公司条件允许的范围内，也无不可。

由于创业公司的业绩具有不确定性，所以股价通常具有主观性且定价机制不透明，但要注意定价不宜超过真实股价过多。如果是无偿赠予核心人员的干股，则要考虑人员中途离职的问题。

目前市面上流行的协议时间在四年左右，因为时间太长会造成过度的压力，太短则起不到激励作用。

购买股份的资金最好由被激励对象出，因为这能将被激励对象与公司利益捆绑，有助于发挥被激励对象的积极性，同时缓解公司的资金压力。

“微创新”的力量

《囧记单词》创始人欧阳丹专访

记者 / 陈博

因为对教育的热情，欧阳丹与朋友们共同组建了“超有爱”团队，打造了拥有百万用户的应用《囧记单词》。欧阳丹说，《囧记单词》在英语教育上的创新来自不断的学习和积累。

“超有爱”团队

“四个普通青年编程序，三枚文艺青年做设计；二名外国青年写例句，一双靠谱青年在编辑。梦想与有爱的你一起，把教育变得更有趣。”

以上这段非常押韵的句子是《囧记单词》开发团队，成都超有爱教育科技的自我介绍。就让我们从这段话开始，了解这支“超有爱”的教育应用开发团队吧。

被称为“囧掌柜”的欧阳丹，是该公司的CPO，一位介于而立与不惑之间的工程师，曾做过几年的C++、Ruby程序员，后来开始带项目做管理，也从事过市场方面的工作，同时学过设计，写得一手好Keynote（苹果PPT软件）。自从有了儿子之后，就非常关注教育，恰逢移动互联网风生水起，遂拖上一帮有爱的青年出来创业。CDO鲁珂在大学本科时获得了计算机和法律双学士学位，又获得了企业管理的硕士学位，他的第一份工作是市场营销，而现在却在囧记画漫画做设计，现在所有的“囧记”形象、日常搞怪微博，均出自他手。CSO陈彬是个狂热的App体验者，曾把工资一半交给了App Store，他是好客的沙发主（couchsurfing），曾混迹北京某著名地图公司，现在负责“囧记”所有产品的交互设计。“囧记”的设计师有两位，一位是拥有自己设计工作室的成名设计师、插画师文静，另一位是在美国留学多年的日本人萨鲁，他为了躲避核辐射来到中国当了陈彬的沙发客，阴差阳错成了“囧记”的一名设计师兼编辑。除此之外，主要成员还有为“囧记”配音的英国小伙金仔，以及奎奎和强强两位工程师。

欧阳丹说，开发者组队创业，最好还是同事、同学、朋友关系，创业合伙人的高信任带来的低流程成本，是小团队对抗大团队的小型核武器。

改变教育的决心

欧阳丹在寻找新的创业方向时，在豆瓣看到了一个用“囧囧的图片”方式学习英语的活动，他用了“虎躯一震”一词来形容当时的感受。在他看来，用这种方法，很多单词都可以一次记忆成型，是一种很好的帮助大家从Ebbinghaus【注：德国心理学家H.Ebbinghaus研究发现，遗忘在学习之后立即开始，而且遗忘的进程并不是均匀的。最初遗忘速度很快，以后逐渐缓慢。他认为“保持和遗忘是时间的函数”，并根据他的实验结果绘成描述遗忘进程的曲线，



公司的白板上有每个团队成员的卡通形象

即著名的Ebbinghaus记忆遗忘曲线】中解脱出来的方法。于是他在第二天就制作了模板放到当时刚刚兴起的微博上去测试大众对它的认可程度，而这个测试的结果就是令《囧记单词》的微博粉丝超过了40万，超出了他们的预料。

欧阳丹的儿子昂昂4岁时，曾有一次问他："爸爸，为什么大组的小朋友有作业做，而我们没有"，欧阳丹惊诧之余意识到一件事：对于小孩来说，玩就是学，学就是玩，玩是他们学习的方式，但一切都不知在何时被划下了一道使其对立的分割线。欧阳丹非常希望未来的小孩可以在课堂上开心学习。他认为，作为一名创业者，强大的团队和有爱的团队文化以及想改变教育的决心，才是他们的立命之本。

既然决定要做这类产品，那么就一定要对同类产品进行研究，基本上国内外背单词学英语的软件，他们都下载来看看有什么值得借鉴的，连英语学习类的图书都买了上百本。

比较的结果是，他们没有发现任何类似囧记的产品，虽然其中有一些书籍是以搞笑为卖点的，但在"囧记"团队的眼里，他们的HHP【注：网络用语haha point，即笑点】太低了。为了挑战更高的HHP，他们大量观看搞笑视频、广告，买来大量书籍充电，系统学习搞笑，举行头脑风暴，让搞笑成为一件"天天见"的事，以至于"囧"文化深入骨髓。

"云平台"必不可少

虽然这是一支"超有爱"团队，但成员们在产品上还是会存在分歧，不过因为大家都有着共同的把教育"有爱化"的目标，所以这些分歧并不严重，反而经历了越多分歧，大家就越明白教育未来到底应该是怎样的，"囧记"成员们把经验总结成名为"学生为王"的三条定律。

- 学生学不好，99%是因为教的水平不够。
- 教学方式一定要"有用"、"好玩"一个不漏。
- 教学水平够高，还学不好的话，一定是因为每个人的天赋不同，说明他不适合这个方向【注：推荐阅读TED超级演讲者Ken Robinson爵士的书《让天赋自由》】。

而在产品开发中，最大的难题就是《囧记单词》的制作成本偏高，每张卡片的制作成本包括人工、图片版权购买等费用超过60元人民币，如果要完成把所有常用单词都覆盖一遍的宏伟计划，会让这支不肯在《囧记单词》中插入广告的创业

团队不堪重负。幸运的是，他们获得了经纬中国移动互联网投资团队的信任和支持，欧阳丹认为，“囧记”在微博上的影响力，也为他们的融资提供了优势。

《囧记单词》的用户分两个群体，微博用户约有40万，App下载量已接近百万级别，最早“囧记”是把服务器架在Amazon的EC2服务器上，由于《囧记单词》是内容下载型App，所以也会大量用到S3的存储服务。目前大部分是不用计算的静态内容，一朵小云服务器用缓存就完全撑得住。所以后来国内的云服务商们，包括盛大云和阿里云相继开测，“囧记”就顺势把服务器回迁，部署在阿里云上。欧阳丹表示，选择阿里云的理由很简单，因为目前他们学习AWS学得最好（20%的核心功能都实现了），服务态度也不错，而最重要的是服务器在国内，而AWS的服务器，离中国最近的就只有新加坡以及东京中心，速度都不算理想。

“微创新”更有效

目前的移动互联网，充斥着大量同质化产品，“创新”是被开发者们大量提及的词汇，《囧记单词》可以算是其中一道靓丽的风景。欧阳丹说，创新的条件永远都有，并且在创新和模仿之间没有明显的分割线，《囧记单词》在业界的创新，是基于对网友智慧的借鉴创造出来的。欧阳丹分享了他最近读的一本书——《科技想要什么》里面的内容：“自然界优先选择的进化是微创新，每一代1%的创新和进化，积累几十代就是翻天覆地的变化。”所以对开发者来说，最大的限制是对“创新”的理解，以为“创新”就一定要做出翻天覆地的变化，但其实往往这种“翻天覆地”的创新，是不被自然界选择的。纵观科技发展史，灯泡被“发明”6次，进化论被“首次发表”3次……其实太多人想到甚至做到了，如果你有一个疯狂的想法，那么只需在借鉴现实的基础上，把想到的做到最好。

当提到未来的发展思路时，欧阳丹给我讲了个故事：有一次，他回母亲的老家去玩，见到了很多儿时的玩伴，他们子女很多都面临升学的问题。



最大的难题就是《囧记单词》的制作成本偏高

欧阳丹在聊天中了解到，他们中的大部分，在争夺县城唯一一所的最好中学的座位未果之后都将只有一条路，如同他们的父辈一样待在老家生活下去。他们中的每一个都很聪明，并且和他们的父母一样勤奋，可是能够接受到的教育机会却非常少。在欧阳丹看来，在未来5～10年内，随着千元以内的智能机、平板的进一步降价，基础教育的公平性，将会依托移动计算的浪潮得到改善，而他对于能够投身甚至推动这么一场变革感到无比兴奋……而《囧记单词》还是一个小应用，能够做好一款产品，解决好一个问题就不错了，至于是否平台化，还要看用户需求的变化。

记者后记

长久以来，中国互联网一直饱受“创新”一词困扰，在很多人眼里，“创新”需要具有颠覆性的创造力，但任何新生的产品内容或形式，都是不断吸取外界养分，量变到质变的过程。大多数公司的成功，都不是基于某一项创意，而是在前行路上，不断碰壁、转型、进步，厚积薄发，然后由某个创意点引爆的结果。所以，那些苦于没有“创意”而停下脚步的开发者们，不要执念于此，“创新”永远源于实践。

如何在 20小时内开发一款应用

文 / Govind Chandrasekhar

在本文中，作者将介绍由三人组成的团队如何在20个小时内开发一款iPhone应用程序和一款Android应用程序，这两者均由Node.js提供后端支持。

传统的应用程序开发流程包括集思广益、形成想法、验证、功能选择、设计迭代，以及最后的开发工作，这一连串的事件通常需要至少一个月的时间才能完成。这一次，我们决定在一天之内便完成从想法到应用开发的过程。当然，我们在20小时内所创建的应用程序虽然功能丰富，但其设计、安全性和可靠性还远远不及发行版的质量。本文除了记录我们的经验以外，还希望能够为那些长期以来一直有想法却迟迟未开展应用程序开发工作的朋友提供动力。

犯罪预警应用的开发时间表

第0～2小时

我在一个周日的下午一点与Sivamani会面。在见面之前，我们已经大概讨论过要开发一个iOS应用程序，帮助人们避免成为犯罪的受害者。我们一直以来都是攻击和破坏行为的受害者，所以希望建立一个预警系统，能够在人们无意进入犯罪高发区时通知他们。我们决定由该应用监控用户的位置，显示其对位置危险等级的评估，并列出该地区最常见的罪行。这样，用户能够避免汽车受损和人身攻击的危险。通过所提供的犯罪报告热点图以及定位系统，用户还能够根据自身的判断来调整地图上的热点。在前两个小时结束时，我们已经列出了想要的所有功能，并确保我们意见一致。

第2～5小时

我们中的两个人开始在线下收集原始数据，大约花费一个小时确定了几个数据源。这些数据源包括纵火、抢劫、攻击、蓄意破坏和性犯罪报告。使用Semantics3的Web收集程序，我们能够在半个小时内掌握数据，随后再用半个小时整理数据并将数据移植到SQL上。我们决定先专注于加州地区，因为这一天的目标是启动产品并使之正常运行，而不深入其他工作内容。同时，Sivamani开始启动iOS应用程序的基础架构工作，我也开始着手API方面的工作。

第5～9小时

API会接收来自应用程序的经度和纬度坐标，并返回该地点“危险等级”评估、该评估中的“置信度”以及进行评估所使用的原始数据的节流清单。我需要花费数个小时用Restify和Sequelize搭建一个MySQL ORM基本服务器，确保代码保留异步调用需要再花费两个小时。

同时，Vinoth已经使用JavaScript实现了一个算法，该算法考虑了每个罪行的地点和性质以及与该数据相关的源数据，其中包括罪行发生的日期和时间。Sivamani已完成了应用程序大部分功能的架构，包括定期轮询GPS和设计要显示罪行统计和详情内容的页面。该应用程序能够与我们为测试所建立的模拟静态API进行通信。

第9～14小时

我们本可以通过SQL查询的内嵌移动Haversian距离功能实现地理搜索，但显然，这不是长久之计。因此，我们改用Sphinx search，在唯一可用的Node-Sphinx连接器Limestone上努力了数小时后，终于使Sphinx地理搜索运行起来（支持地理搜索的Limestone代码很快将启动，目前的版本无法提供足够的支持）。

同时，Vinoth致力于将来自多个源的数据合并在一起，尤其是合并性犯罪和与一般犯罪的数据。然后，他将聚集算法合并到Node服务器，调整JSON响应以满足应用程序的要求，最后得到并调用API。此时，Sivamani已经完成了应用程序的地图和显示详细内容的视图，并使之与API协同工作。

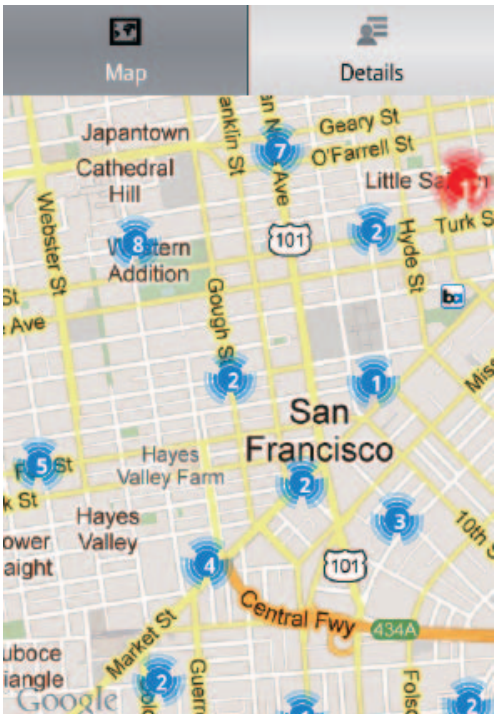
第14～20小时

同时，Sivamani想到了一个有趣的功能。因为儿童很容易不慎进入危险区域，并且不可能想到用预警应用程序获取信息，因此他添加了一个“儿童模式”。从本质上来说，该应用程序将在孩子处于危险时自动向孩子的两个紧急联络人发送短信。Sivamani通过集成Hoiio API来实现这个功能，不久后，他调整了该应用程序的图形和色彩，并添加了精巧的修饰。

当Vinoth优化了API和测试算法时，我决定开始编写Android应用程序。我以前已经编写了多个应用程序，这些应用程序都需要多个地理位置和映射；因此，使用我以前编写的一些代码，便能够搭建基本框架。此后，我着手于快速复制Sivamani在iOS上所进行的工作。在第20个小时的最后，完成了Android应用程序，虽然里面没有Sivamani所集成的一些很酷的功能。

要点

心动不如行动！应用程序的开发在一开始就是件非常繁重的任务。开展工作的一个好方法就是开始编写代码。建立起整个基本框架是当务之急。在开始这一天的工作之前，我们对Node.js API、Sphinx搜索、iOS地理位置、SMS API等不甚了



这款应用可以显示用户所在位置的危险等级

解，但现在我们已经比较熟悉这些概念，足以设置该功能的知情期限以及进行熟练的功能选择。

在开始编写代码之前需要考虑好你的基本设计方案。繁杂的设计将很难修改，并且通常会造成代码冗余。

在创建应用程序时记下该功能的“待办事项”清单。这包括不属于最小工作原型类别的其他功能或安全/设计/可扩展性/跨设备考虑事项。例如，我已经写下在下一个迭代过程中为Android应用程序构建九格地图。

如果你的应用程序依靠预生成的数据，则专注于子集而不是试图立刻准备好所有数据。首先要整合最棘手的数据集，其次再整合次要的。这种开发模式可能不适用于所有类型应用程序的开发，特别是那些图形密集型应用程序，因此请运用自己的判断力来识别我们的经验和你的需求之间的相似之处。

本文作者Chandrasekhar来自一个为开发者提供移动数据解决方案的团队Semantics3，其网站为semantics3.com。

水果忍者

基于返璞理念和前瞻营销的终极荣耀

文 / 郑金条

尽管拥有好的创意无疑是占领市场的首要条件，但返璞归真的游戏设计理念和前瞻性的营销策略，是《水果忍者》能够成功的两个不可忽视的因素。

随处可见各种对Halfbrick Studios旗下《水果忍者》的歌颂，无论是来自玩家的游戏体验还是来自专业评述的肯定，玩家对这款游戏的推崇不亚于《愤怒的小鸟》。

早先有消息认为老少咸宜的《水果忍者》在全球用户市场的下载量已经超过1.4亿次的高度，如果细分到App Store，《水果忍者》始终都是付费下载游戏榜的前十名。Mobclix曾以《水果忍者》为例来分析手机端在60秒内究竟能够产生多大的价值，而结果显示，《水果忍者》每60秒可以实现80次下载，有250000粒水果因此被玩家切碎（数据截止日期为2011年11月）。

Pocketgamer在2012年全球顶尖的手机游戏厂商评选中将Halfbrick Studio从2011年的第7位提升为第2位，仅次于Rovio，领先诸如EA Mobile、Gameloft、Chair Entertainment和Popcap等几家营收大亨。

即便是在Sean Ryan（Facebook）并不看好的偏单机版Facebook游戏市场的境况下，仍然像先行者Popcap旗下的《Bejeweled Blitz》（1080万MAU，310万DAU）一样风靡用户市场。 Appdata的最新数据显示，《水果忍者Frenzy版》拥有760万的MAU，比上周净增17%（DAU为110万），已接近另外一个经典游戏《愤怒的小鸟》（MAU为1280万，DAU为300万）的用户表现。甚至知名游戏网站Games已经将《水果忍者Frenzy版》归属于年度最佳的社交游戏行列，排名第6，甚至领先于Zynga的《Empires & Allies》、《CastleVille》和Digital Chocolate的《Zombie Lane》。

而这可以用相当纯粹的数据和荣耀进行佐证。

- 苹果在App Store应用下载达到250亿时发布了迄今为止最为热门的付费游戏，《水果忍者》仅次于《愤怒的小鸟》而位居第二。
- 《水果忍者》是第十五届Annual Interactive Achievement Awards年度最佳休闲游戏（游戏移植版《水果忍者Kinect》，击败了Halfbrick Studio旗下的另外一款游戏《Jetpack Joyride》、Disney Mobile旗下的《Where is My Water》和超级个人开发者Andreas Illger旗下的《Tiny Wings》。
- 《水果忍者》是微软Xbox Live游戏前20名，排名第2，仅次于Rovio旗下的《愤怒的小鸟》（领先于PopCap旗下的《植物大战僵尸》）。
- 入围了第十五届Webby Awards最佳掌上游戏大奖（最终奖项的获得者是Rovio旗下的《愤怒的小鸟》和Zynga旗下的《Farmville》iPhone版）。
- 2011年金摇杆奖Golden Joystick Awards的提名游戏。



《水果忍者》的成功归功于产品设计上的极简化理念和用户行为渗透分析

■ 《水果忍者》是亚马逊App Store和Android Market最流行的付费游戏之一。

返璞归真的游戏设计理念

事实上，《水果忍者》是大量简单闲趣游戏成功的缩影，诸如最近异常火暴的夫妻档工作室Imangi Studios旗下游戏《Temple Run》和Disney Mobile旗下游戏《Where is My Water》都有相似的特征：简单、闲趣、简短关卡、家庭娱乐性。

这种特征和受欢迎的共性在Halfbrick Studios旗下另外一款游戏《Jetpack Joyride》得到了验证，虽然没有《水果忍者》那么庞大的下载基数（也是2000万级别），但在用户市场和专业评论市场获得了双重赞誉：《Jetpack Joyride》是Appolicious版年度最佳iOS游戏；USA Today的Game Hunt认为《Jetpack Joyride》是年度10大游戏之一（跨门类评选）；Appolicious认为《Jetpack Joyride》是年度仅次于Chair Entertainment旗下《Infinity Blade II》的最佳iOS游戏；Pocket Gamer Awards 2012奖项评选中，《Jetpack Joyride》斩获了最佳Action/Arcade游戏、年度iPhone/iPad最佳Touch游戏和年度最佳游戏三项大奖。

这种连续性的成功，可能要归因于Halfbrick Studios在产品设计方面的极简化理念和用户行为渗透分析。

Mashable在提名《水果忍者》为2010年度最佳游戏时，提到了该游戏最吸引人的部分：没有枯燥和烦琐无聊的操作指南，没有累赘的玩法设置，对于玩家来讲他们只有一个极其简单的切削水果任务。

Keith Stuart也认为《水果忍者》的成功在于其在游戏UI（布局和功能）和新手引导（手势指引，游戏界面中一只意象手提示玩家需要以滑动才能进行游戏）方面便捷地呈现了菜单系统和交互指南，完全融汇了游戏设计中的超级理念：好的设置让玩家甚至不会察觉到自己正在学习一些必要的内容而自然切入（以手指划切水果，并随时注意提防可能出现干扰的炸弹设置，如果玩家能够完成一些高难度的切法还能够获得系统的额外嘉奖，甚至有评论者认为这是基于指尖的巫术）。

Halfbrick营销总监Phil Larsen和公司董事长Shaniel Deo在对外专访提到该游戏的设计理念时都认为最终的创意效果都是历经不断地向外借鉴和不断更迭的游戏原型才逐渐磨砺出来的，Phil Larsen甚至认为基于既有市场和用户双重剖析包括现有创意的原型实践，他们基本上就能够预判

一个游戏项目未来的成功潜质。

前瞻的游戏营销策略

事实上，如果我们潜下心来看待诸如《愤怒的小鸟》和《Cut The Rope》的成功，都有相似的市场营销进程，《愤怒的小鸟》首先在芬兰有了小规模传播，《Cut The Rope》在莫斯科游戏先有小规模的传播，然后再借助发行商（Chillingo）和苹果App Store的排位优势崛起并牢牢把控排行榜的前端位置，占据着App曝光度的主动权。

或者说，至今在App端能够成功的游戏都没有无缘无故的成功，除了游戏本身的质量获得认可能够引发口碑效应之外，早期游戏的用户基数累积也至关重要。按照Halfbrick的营销说法，那些运营不善的公司往往在万不得已的情况下才舍得花费营销经费，而事实上在既成困局下，这种受迫式的亡羊补牢效果并不显著。Shaniel Deo就认为前瞻性和激励用户自发分享的营销是游戏企业获得成功的关键要素，而一些未能俘获市场的游戏甚至在游戏的开发后期都没有相关的营销筹划，甚至还只是固执地认为好的游戏能自动获得喜爱进而有超级下载数据。

尽管好的创意产品无疑是市场的首选，但仍然会有一部分成功因素隐藏在技术以外，特别是开发者如何抓住市场的时机，如何去运作和布局整体的营销行为，或者与平台方有什么样的特殊协作关系，这些都是透明市场中的阴影，而正是这些阴影守住了产品的另外一个秘密。

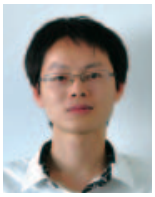
早先我们曾对这一情况进行过论述认为问题可能出现在：看起来没什么差别的画面、雷同的玩法、相似的营销策略、相同的渠道、再加上如果开发者讲不清自己的故事、理不出游戏的核心，卖点就有可能被迫隐形。甚至一些基于该创意的雷同产品都可能为原创带来冲击，比如对《水果忍者》进行模仿的《切水果》、糅合《水果忍者》创意的《Cut The Birds》，或像《Veggie Samurai》；比如对《Temple Run》进行模仿的《Temple Jump》或者《Temple Gun》之类的游戏，都有相当火暴的下载市场。

事实上在现在这个阶段，提高曝光率已经成为一场持久战，产品营销和企业的品牌诉求并行，越来越多的游戏公司试图超脱游戏的疆界，化身成为超级娱乐品牌。

如果从App角度营销域值的变化，不管是Rovio还是Halfbrick都认可了社交网络（特别是Facebook和Twitter）对产品直面用户营销的价值。作为开发者，对自己游戏产品的特征和用户需求假想将比其他环节更加了如指掌，不需要凭空捏造需求，也不需要在各种渠道之间角力去搜寻用户，用户本身已经真切地布局在交互媒体上，并且更加真实（各种关系链和情感投入）和主动。

和传统行销依托广告公司布局媒介渠道的方式相比，社会化营销几乎已经消弭了让广告主潜藏在包装表演背后的那种隔阂感。Shaniel Deo分析了《水果忍者》初步发布时的状况，那时他们相当依赖社交平台进行用户的口碑营销，让用户以发布积分和成就的方式怂恿更多的用户接力挑战。这种思路比较暗合病毒传播的第二层面——基于用户的认同感和喜悦的相互分享，这种分享带有期待共进的成分，有相当强的建议性。

Shaniel Deo在早期的第二个方式是与媒介的联动，而这种联动需要形成机制，包括如何去传递游戏的进展、游戏的理念或者定向的媒介群和用户群。尽管现在看起来每家游戏开发商都在走相同的营销步骤，但试探深浅和对媒介与营销的传播力，包括对自己产品延伸力的理解都具有相当的差异性，需要从受众和产品本身的双向界定中解析出有价值的环节，并能够成功预热一个市场，至少让潜在消费者产生某种期待感。



郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

责任编辑：陈博（chenbo@csdn.net）

从 Enyo看移动 Web应用框架设计

文 / 程邵非

Enyo是一个来自Palm的Aries项目的JavaScript开发框架，它所关注的方向是一般的程序框架所关注的封装性、复用性和可维护性。

历史

Enyo是随着webOS发布的开发框架。

Enyo 1.0发布于HP TouchPad的webOS之上，它是依赖于WebKit的开发框架。因为一些历史原因，Enyo 1.0的授权仅限于webOS中使用，并没有使用开源协议。

Enyo 2.0则是一个跨平台的框架，它现在已经在github以Apache 2.0协议开源。

如无特别强调，本文后面所述的Enyo皆指Enyo 2.0框架。

核心概念

如Enyo网站主页强调的，Enyo的核心是一种封装模型，让使用者能够更容易地复用和维护。在这一点上，Enyo更倾向于一个传统程序设计领域的框架。它所关注的方向是一般的程序框架所关注的封装性、复用性和可维护性。

与Enyo相比，一些流行的JS框架设计的核心思想则显得不那么“程序员思维”，比如jQuery的设计理念是：Write less, do more。

jQuery在这一点上颇受一些程序员背景的工程师诟病，但恰恰因为这样，它讨好了大量设计师和有设计背景的前端工程师。

Enyo的设计理念是将程序分解为一个个的Component（组件），Component又能够由其它的Component组成。Enyo是一个立足于UI的框架，但它的Component理念并不限于UI组件。

Enyo这样的设计非常适合较大的工程，这跟传统的Web前端项目理念有些不同。很难想象一个网页会由很多工程师共同完成。而Enyo认为基于Enyo开发的应用程序，可能需要更多工程师来进行合作。

总体而言，从Enyo的整体设计看，它对于移动开发程序员和程序员背景的前端工程师比较友好，而与之相对，它对于设计师背景的前端工程师则不那么友好。

Enyo的设计目标:Web、移动、应用

Enyo作为应用程序基础框架用于webOS上，这决定了它的设计目标离不开Web、移动和应用三个关键词。

毫无疑问，webOS选择HTML+JS的开发方式很大程度是看中了已有的网页前端开发工程师资源。经过多年的发展和磨合，现在网页开发已经是一个非常成熟的技术环境了，这其中有着由设计师、工程师、产品经理等不同人才组成的完整的生态系统，大部分企业的开发流程也已成熟。



Enyo主要分为五个模块：Kernel、AJAX、DOM、Touch、UI

尽管拥有JavaScript这样图灵完备且具有相当抽象能力的编程语言，网页与应用之间仍然有着不小的差距，而Enyo的设计目标之一就是尽量弥合这个差距。

为了与同类产品竞争，Enyo还必须要提供能够快速上手的开发能力，以及对硬件的访问能力。而这些恰恰是传统网页所缺少的功能。

Enyo的编程模型中基本抛弃了HTML语言本身，而主要使用JavaScript和DOM API来编写图形界面。Enyo选用JSON来代替HTML作为界面描述语言，这样既保持了方便的程序访问性，又保证界面编写难度不会过高。

JSON描述界面这种做法在传统的JS框架中有成功的先例，其中最为著名的当属ExtJS，Enyo选择这一做法很可能借鉴了ExtJS的成功经验。

当然抛弃HTML这一做法实际上意味着Enyo放弃了大批Web开发者以及围绕Web开发的工具。Enyo的这一选择可能并非明智。

因为webOS系统主要用于移动设备，所以Enyo的设计也必须更多考虑移动设备。Enyo提供了专门为触屏设计的Scroll组件，这样可以使webOS设备能够轻易地做出接近iOS和Android设备的触摸滚动效果。

Enyo的架构

Enyo主要分为五个模块：Kernel、AJAX、DOM、Touch、UI。

- Kernel模块为所有模块所依赖，其中主要包括对JavaScript语言本身的扩展和Enyo的Type系统。
- AJAX模块实际上不仅仅是AJAX功能扩展，还包括JSON等AJAX相关的功能扩展。
- DOM则是对HTML DOM API的简单封装和基本手势的支持。
- Touch实际上包含了Scroll和Touch功能的结合，touch提供了各个层次的Scroll逻辑，既能够直接使用Scroller，又能用更为底层的ScrollMath和ScrollStrategy来定制自己的Scroll行为。
- UI模块是Enyo提供的基本控件，这些控件的主要作用是生成对应的HTML代码。

加载系统

Enyo的主要模块之外，有一部分Boot模块可以用于加载Enyo的脚本以及注册全局命名空间。

这里Enyo提供了一组简单的包管理系统，包括加载包、包依赖关系，以及样式表单文件的管理。

虽然很少用到，但这部分功能也暴露给了开发者，通过enyo.loader可以实现加载代码包并且处理依赖关系。

OOP系统

Enyo实现OOP的方式是Type+Component模式，这基本可以认为是Class和Object的别名。

Enyo的Type系统是Enyo提供的一种类型系统，一切Enyo框架内的组件都必须遵循Type系统来注册自己。

Component系统则是与Type系统密不可分的，任

何Component都必须有自己的指定Type，Enyo正是通过这种方式保证模块化的。

以下是来自Enyo官方网站的一个注册类型实例。

```
enyo.kind({
name: "Hello",
kind: enyo.Control,
components:
[
{name: "hello", content: "Hello From
Enyo", ontap: "helloTap"},
{tag: "hr"}
],
helloTap: function()
{
this.$.hello.addStyles("color: red");
}
});
```

值得注意的是，Enyo可以简单地将几个Component组合，形成新的Component类型。

Enyo注册类型之后，则可以直接通过new来创建Component。

在kind函数接受的JSON描述中，kind属性起到了等同于指派父类的作用。

扩展API

如同每一个JS框架做的那样，Enyo也有对XHR、Cookie、事件、异步等一些HTML DOM API做了扩展，这样可以减轻Enyo的开发者的负担。

比较特别的扩展是Enyo对于手势的封装，Enyo提供了10种手势：up、down、tap、move、enter、leave、hold、release、holdpulse、flick。这些基本涵盖了所有的单点触控手势。

scroll系统

在Enyo中最简单的支持滚动的方法是直接使用Scroller和TouchScrollStrategy，Scroller提供了基本的滚动行为，TouchScrollStrategy提供对手势的基础支持。

实际上Enyo暴露了不同层次的API，最底层的是ScrollMath类型，它定义了一个效果比较良好的滚动所用到的数学和物理常量以及算法。

ScrollStrategy则是一个抽象类，它规定了手势与Scroller之间的关系，如果不使用原生TouchScrollStrategy的话，则可以自由定义事件和滚动的触发关系。

思考

Enyo作为一个已被正式商用的移动应用开发框架，无疑是值得后来者学习的，包括其他基于Web平台的操作系统、试图为hybrid服务的Web框架以及类似PhoneGap的HTML应用壳程序。一些Enyo受到诟病之处，也值得参照和避免。

Enyo中可以总结一些移动应用框架的设计原则。

尽量保持原生风格

在框架设计上，采用颠覆性的风格是危险和得不偿失的，应当尽量贴近原生API的命名风格和设计风格。

提供多层次的API

既给初级用户提供能够简单使用的API，又给高级用户提供足够灵活的API，当两者矛盾之时，可以通过暴露不同层次的API来实现。

节省命名空间

如非必要，尽量不污染全局命名空间。

支持触屏设备

现有局面下，移动框架的主要竞争焦点在于触屏的支持，尤其是手势和滚动效果很大程度决定了框架的优劣。

考虑开发者多样性

Web App的开发者包括传统的前端人员和一部分移动开发工程师，设计框架时过于倾斜将会导致一部分开发者流失，尽量提供给用户多种选择更为合理。



程劭非

盛大创新院研究员，目前在盛大创新院前端项目组。负责盛大电子书Bambook的排版和格式开发。之前曾在微软负责Windows CE系统上的IE开发。

责任编辑：陈博（chenbo@csdn.net）

从架构角度看移动 App之争

文 / 曹文斌

Native App和Web App孰强孰弱？Web App是否在将来能取代Native App？这一直是移动互联网从业者讨论的话题，本文将从架构的角度来进行分析。

还记得C/S和B/S吗

类似的技术趋势之争实际上已经有很多了，我们不妨花点时间回顾一下十年前的C/S和B/S之争。

20世纪90年代，C/S技术成熟并得到了快速发展，即系统是由客户端和服务器两部分组成。客户端部分的主要任务是提供用户交互的界面，控制应用程序的逻辑。服务器负责有效地管理系统的资源，尤其是对安全性要求较高的数据库管理和访问控制等。

在C/S架构中，对产品开发的要求是两端都需要大量的开发工作。客户端主要是针对Windows等用户操作系统平台进行开发，使用大量与操作系统相关的API，开发工作量大，版本更新困难。服务器端主要是进行数据库管理方面的开发工作。因此，开发人员会分成两类，基本上没有交集。在系统维护上也比较复杂，最难的就是客户端的版本控制和升级工作，给维护工作带来很大的工作量。另外也不适合移植，如果用户要求要支持Linux系统的客户端，那就只能投入更多的客户端人力重新开发一套客户端软件。

随着三层C/S结构、浏览器、脚本语言、Web技术的发展，出现了B/S架构，即系统是由浏览器Browser和服务器Server两部分组成。浏览器采用用户设备上自带的各种浏览器，这样在用户设备上无须再安装客户端软件，也无须为客户端软件支付开发成本。服务器端采用三层或多层结构，利用脚本语言等形成以网页形式提供的用户交互界面，然后与业务逻辑和数据库进行协作。显然，相比C/S结构，B/S一方面可以减少客户端的开发和移植工作，节省客户端的维护费用；另一方面由于表示层、数据和业务逻辑都在服务器端，可以自由地进行系统维护和升级工作。

但B/S结构也有不足之处，比如受网络制约，同时安全性难以控制。另外B/S也不能满足所有的用户需求，比如浏览器受安全性限制，不易于操作用户设备上的更多资源。于是浏览器插件被引入进来，通过浏览器插件，一些需要使用用户本地资源的功能得以实现。

直到今天，Web技术已经发展到2.0时代，Web页面的展现效果已经非常丰富，但C/S和B/S两种架构依然同时存在，二者的优劣都相当明显，谁也无法替代对方。

Native App

Native App架构的客户端中包含了界面展现、业务逻辑、数据模型三个部分，如图1所示。通过网络访问服务器端的各种服务。以手机导航软件为例，在手机上会使用相应操作系统的API开发界面，然后调度业务逻辑层的进行功能实现。当需要展现地图时，如果手机上没有地图数据，就会使用经纬度坐标访问服务器上的地图服务，从服务器获取地图数据，解析后再通过界面绘制出来。在这样的交互下，服务器端的服务独立性较强，其业务形态完全是由客户端控制和调度。

这与C/S结构是一样的。由于使用了客户端的众多特性，所以客户端的表现力相当丰富，性能较高，用户体验也比较出色。但劣势也比较明显：客户端的开发工作量大，逻辑复杂；客户端不利



图1 Native App架构图

于移植，必须针对不同的操作系统进行单独的适配开发工作；客户端软件升级和维护困难；服务器端要支持多客户端，难于扩展。

Web App

Web App基本上对应了B/S架构，如图2所示。客户端不再需要开发工作，退化为使用内置的浏览器与用户交互。服务器端以HTML5、CSS3、JavaScript作为前端供浏览器展示，使用其他服务器端技术实现业务逻辑和数据库管理。相对于C/S结构，其最大优势在于无客户端，不需要专门的开发和移植工作。即使使用WebView控件的客户端，也易于开发和移植。另外的优势就是由于代码全部在服务器端，所以易于扩展和升级。

同样，Web App的劣势也比较明显：服务器端的开发工作量大，逻辑复杂；界面响应稍慢，性能弱；难于使用设备的特性比如传感器、GPS定位、本地文件系统等。



图2 Web App架构图

Hybrid App

在目前Web App受限于浏览器或前端技术未标准化的情况下，可以进行组合应用，这种应用也被称为Hybrid App，结构如图3所示。在这种结构中，对于侧重性能、体验、设备特性、本地数据管理的部分，采用Native的方式，其余的部分采用Web方式，通过这种结合，可以有效地形成互补。这就像在B/S结构上增加浏览器插件一样，对二者进行融合。

现在采用这种融合方案的应用正在日渐增多，既可以发挥本地应用的优势，又能有机地整合后端资源。对于需要本地化的部分，继续采用操作系统的API接口进行客户端开发，对于适合Server端的部分，通过嵌入WebView等浏览器控件进行实现，也可以采用像PhoneGap这样的框架来消除网页展现中难于使用设备本身特性的缺点。

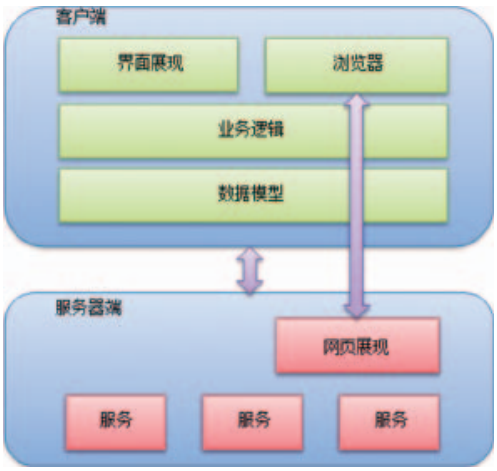


图3 Native与Web融合架构

结语

移动应用并不是非此即彼，必须充分认识到二者的特性，然后灵活地应用在自己的产品架构当中，才能更好地平衡软件的结构。



曹文斌

百度移动互联网事业部架构师，主要负责制定手机客户端产品的技术路线和实施方案。

责任编辑：陈博（chenbo@csdn.net）

Chrome for Android 对 Web App发展的影响

文 / 刘铁锋

新的JavaScript引擎、硬件加速、HTML5 Canvas的性能提升以及新的网络协议，将使Chrome内核成为Android平台上的主流浏览器内核。

根据VisionMobile在2012年2月发布的跨平台技术研究报告，在调查的2400多个开发者中，已分别有32%和30%的开发者在考虑用PhoneGap和Sencha之类的技术来解决跨平台的问题。除此之外，投资者也已在跨平台开发工具领域投入了总计高达2亿美元的投资。这充分说明，在Web App的框架和工具领域已经做好准备，相关的工具以及框架将会得到越来越多的发展。不仅如此，海外的一家移动框架公司AppMobi也快速推出了号称提速3~4倍的JQ.Mobi，试图利用速度来让移动设备上有可行、可用的JavaScript框架。当这些框架都逐渐成熟起来时，Web App将会有更完善的开发工具支持。除此之外，还有哪些因素会制约或者限制Web App的发展呢？那就是Web App的运行平台——手机浏览器。手机浏览器对Web App性能的改善将大大地推动Web App的技术发展和实施。

浏览器的发展对Web App的影响

众所周知，Web App的运行平台就是浏览器。运行平台的发展直接制约了Web App的推广与普及。在移动设备发展的早期阶段，由于网络资费过高，上网速度过慢，手机硬件处理能力较差，协议简单、内容单一的WAP网站成为主流。由于WAP基本上不支持JavaScript，所以对于JavaScript引擎的要求也不高。在处理能力比较单一的情况下，浏览器的发展速度也处于比较早期和原始阶段。这种浏览器基本上不能作为Web App的运行平台。

随着智能手机的发展以及处理能力的进一步增强，手机已具备了直接解析PC网站的能力。目前手机的主流配置可以达到1GHz以上的CPU，以及高达512MB的RAM/ROM，而高端的智能手机可以达到双核1.5GHz，几乎赶上了当前PC的配置。而屏幕的分辨率则更可以高达1280×1024，这和当前PC的屏幕分辨率基本一致。也就是说，在智能手机的硬件处理能力上，已经和PC没有任何区别。因此，智能手机的硬件已经为Web App的发展做好了准备。那为什么智能手机并没有对Web App的支持有很大的推动作用呢？原因很简单，智能手机的浏览器技术发展过于缓慢，对于主流采用中转压缩方式的浏览器来说，并没有在浏览器内核上对性能和速度带来任何本质提升，导致Web App很难流畅运行，使得同样的硬件无法达到PC上的同等效果。

从技术趋势、Web App的需求上来说，已经要求智能手机浏览器进行革命。这种革命会推动能够发挥全部硬件潜力、更好支持HTML5、无须转码

直接显示网页的现代浏览器全面取代采用中转压缩省流量的传统手机浏览器。

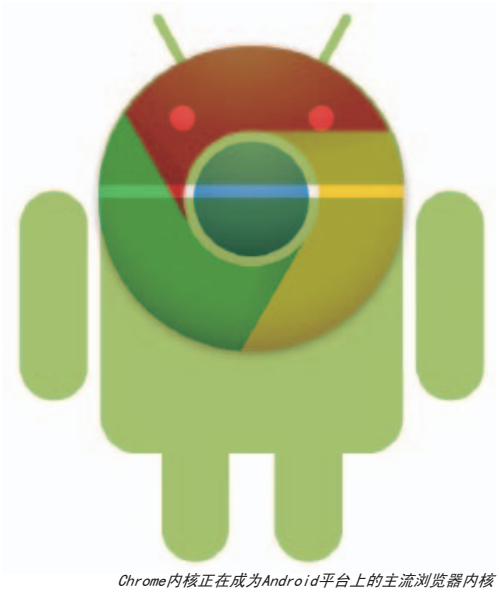
Chrome for Android带来改变

2012年2月7日，Chrome for Android正式登陆Android Market，尽管仅支持Android 4.0，但在短短几周之内就已经突破了10万下载量。尽管相对于Android数千万甚至上亿的出货量来说，哪怕是相对于很多的第三方浏览器的下载量来说，这点下载量几乎不算什么，但如果对比一下Chrome在PC市场上的战绩，就不得不让所有在Android上耕耘的第三方浏览器打起精神，好好思索下一步的发展。要知道Chrome在短短4年之间，已占领了PC市场上18%的份额，并且已经超过了火狐的市场份额。除此之外，在Windows/Linux/Mac上的全线出击，几乎让所有的其他竞争对手感到无力，能想象微软的IE出一个Linux或者Mac版吗？如果对Chrome的来历做深入的研究，就会发现这是Google在扩展其在服务器端优势到客户端的重要一环，不仅仅是通过浏览器占领用户访问网络的入口，甚至Google还推出了基于Chrome的OS，并且已经在市场上发售。因此，也不难想象，就有如微软非常重视Windows一样，Google也是集中了精兵强将在Chrome上进行投入。凭着Chrome超快的渲染速度、JavaScript的执行速度、Chrome几乎成为浏览器上的一个传奇。而Chrome for Android的发布，借助硬件加速、对HTML5的更好支持以及更快的速度必将快速地摧垮所有传统浏览器的优势，使Chrome内核成为Android平台上的主流浏览器内核。

Chrome对Web App的推动作用

那么Chrome将会对Web App带来怎样的推动作用？让我们先看看Chrome改变了什么。

■ **JavaScript引擎**。JavaScript是编写动态内容的基础语言，JavaScript代码的执行速度，直接影响了Web App的响应速度和表现。因此，Chrome选择的第一个突破点就是JavaScript引擎，它重写了WebKit里面所用的JavaScript引擎，创造了V8。



Chrome内核正在成为Android平台上的主流浏览器内核

这种新的引擎号称速度比传统的JavaScript引擎快60%以上。这件事看起来非常简单，但意义重大，这会直接提升Web App的执行速度，但难度确非常之大。如果联想起众多语言，比如Java、Python的创立者都在Google工作，就知道为什么Google有这样的技术实力来做这样的事情。除此之外，把JavaScript引擎在WebKit里面剥离，解决兼容性的问题，这更不是一个简单的工作。因此，在这一点上，Google利用其强大的技术优势，把微软、苹果、火狐等竞争对手甩在了后面。而对于Web App来说，最影响响应性能的动态JavaScript执行，将被Chrome的V8带到一个新的高度。

■ **硬件加速以及对HTML5 Canvas的性能提升**。如大家所知，HTML5是目前非常重要的一个标准，也是指导浏览器进一步发展的路线图。对HTML5特性支持的多少以及好坏，决定了浏览器在下一个竞争阶段所占领的位置。因此，Chrome全面支持了HTML5，尤其是利用硬件加速来支持HTML5的Canvas标签。也就是说，Chrome会比没有使用硬件加速的浏览器快。有数据表明，在Android 4.0上，使用了Chrome之后的Canvas的性能能够提高5～10倍。对于Web App来说，不管是应用还是游戏，都会对是一个巨大的性能提升，并且对所有开发者来说，这为考虑选择Web作为移动应用的解决方案提供了技术基础。

■ **网络协议**。众所周知，HTTP协议是整个互联网的基础协议，所有用户看到的网站都基于HTTP协议来进行数据传输。这就像构建计算机的基石一样，几乎无法更改。

但在Google眼里，这也不是不可能的事情。于是，天才的工程师就想了一个办法，在HTTP协议上做改进。理论上来说，改进协议这件事大家都想干，但协议实际上是包括服务器端和客户端。改了客户端，还有众多的服务器端不能支持。但Google并没有太担心这个问题，因为它就是流量最大的网站之一，自家的服务器快了，就很不错了。它在HTTP的基础之上创建了一个叫speedy的新协议，这个新协议号称比HTTP要快60%。能够想象，访问同样一个网站，别人的浏览器打开要2秒钟，而Chrome只需要1秒钟就可以打开。

这将对Web App的资源载入速度带来巨大影响。因此，尽管目前还没有太多的第三方服务器支持，Google自家的服务器全部启用了对speedy的支持。尽管这一点的普及需要时间，但也为提升Web App资源的加载速度指明了方向。

除此之外，Chrome也是改进最快的浏览器。在短短的4年内，Chrome以疯狂的版本发布速度（目前发展到19.0），彻底拖垮了其他竞争对手，市场份额也节节攀高。而这样的性能改进和速度提升，更为Web App的发展铺平了技术上的道路。

Web App的发展障碍

众所周知，目前的智能手机上，采用中转压缩流量的浏览器依然延续了在2G时代的辉煌，占据了比较大的市场份额，而这些浏览器却成了Web App发展的最大障碍之一。对于Web App来说，中转压缩的方式已失去了意义。中转压缩的技术方式很难适应3G时代以及大屏幕、强处理能力的需求，以及基于HTML5的Web App的发展。随着网络速度的提升，处理能力的增强，中转省流量的技术会被时代所抛弃。而对HTML5支持好、页面渲染能力强、硬件加速能力强的浏览器则会成为主流。Chrome for Android的出现，更是大大地推动了这一进程。相比起当前的默认浏览器以及第三方浏览器，硬件加速、更快的渲染速度、更快的网络访问速度，都会和PC上一样，直接拖垮第三方浏览器。试想微软这样靠软件起家、工程师实力和储备强大的公司都在Google的攻势下节节败退，更何况是寄希望于在现有WebKit基础之上，做点修补工作的自主内核浏览器，会受到最为直接的冲击。不仅如此，反而会进入进退两难的局面：如果不跟随Chrome，自己的改进很难达到Chrome的水平，而跟随Chrome，自己的辛苦努力将会全盘抛弃。这对于任何一个吹嘘内核技术的浏览器厂商来说，都不是一个太好的消息。除此之外，Chrome for Android的到来，会大大拉开第三方浏览器的竞争格局。面对Chrome的更新速度，只有靠进一步改进技术，跟上Chrome的速度，才可能有所斩获。而原有的市场优势，或者过渡时期的技术优势（比如通过服务器中转），都会慢慢丧失优势。而一旦Web App的大规模普及，这些中转压缩的浏览器就成为影响Web App发展的障碍。

结语

智能手机浏览器的发展和Web App的发展是共生的关系。Chrome for Android的到来肯定会对现有Android移动浏览器格局带来很大的冲击，也会对Web App的发展带来巨大的推动。这种冲击几乎是革命性的，基于HTML5的Web App越多，越会促使用户抛弃牺牲用户体验的手机浏览方式。而此时Web App的基础技术框架应该会准备完毕，从而更快地推动Web App的发展。



刘铁锋

百纳信息技术有限公司CTO，W3C标准化组织成员。带领团队开发的海豚浏览器在CNET评选的2011年全球最受用户欢迎Android应用中排名第二。

责任编辑：陈博（chenbo@csdn.net）

Siri详解之输入系统和活跃本体

文 / 张俊林

Siri是iPhone 4S内置的智能语音识别系统，吸引了许多用户的关注。本文将从技术层面详解Siri，主要讲述输入系统和活跃本体两部分内容。

寰球今日，苹果正红。

iPhone 4S无疑是苹果公司的博利杀手及系列产品中的当家小生，除采用A5双核处理器、800万像素摄像头以及全新的iOS 5外，最吸引眼球的还是其内置的智能应用Siri。

Siri并非苹果公司自主研发，最初只是苹果App Store上的一款应用，由Siri公司开发。2010年4月28日，苹果公司收购了Siri公司，传言收购价格在2亿美元左右。不久之后Siri从App Store下架，集成进iPhone 4S。

从纯技术角度，Siri是一套较准确的语音识别系统，也是具有环境感知能力的持续自学习系统，更是一整套集成上述所有优点的新型人工智能框架。本质上，Siri是通过多轮对话来获得用户事务意图的意图识别系统。为了做到这一点，其内部综合了语音识别、自然语言理解、任务控制、意图理解、服务集成等多项技术。

用一句话来概述Siri：Siri能理解用户的语言语义，并经过多轮会话弄清用户的意图后，集成能够满足用户意图的多种服务（内部和外部）来帮助用户达成特定领域的事务。

Siri整体技术框架

从整体上来说，Siri符合云计算的技术架构，图1展示了其物理架构。Siri正常运转需要调用词典资源、计算模型等很多资源，客户端也会存储部分数据资源和计算资源，但绝大部分数据和资源存储在服务器端。客户端接收用户输入并做简单处理。

就语音流来说，Siri会将语音数据传输给服务端进行语义理解和意图识别等操作，并将服务结果在客户端展示给用户。Siri对用户需求的满足，最终要体现在能够集成提供一定功能的各种服务上。Siri提供的所有服务，一部分是苹果自身提供的，但大部分服务都是通过调用外部网站提供的服务API来完成的。

Siri是一个功能繁复的综合AI框架，图2展示了其包含的数据、模型和计算模块。为了更易于理解整个框架，本文将其划分为能顺序执行有先后依赖关系的四部分。

- 输入系统支持用户多种模态的输入，并在内部统一以标注文本的方式表示。

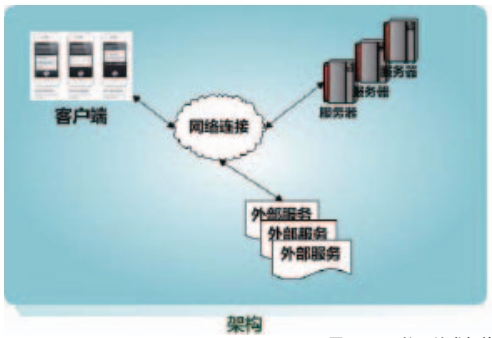


图1 Siri的云技术架构

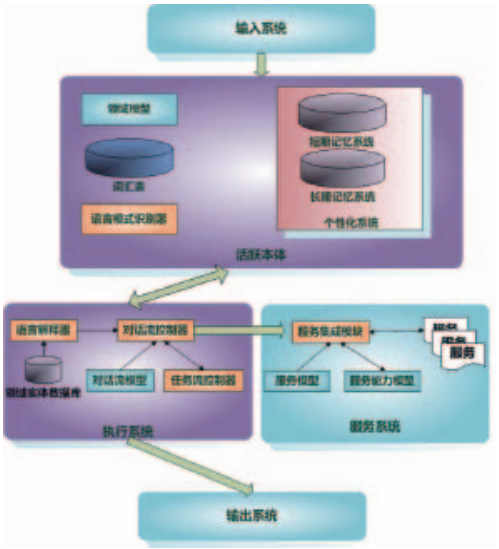


图2 Siri技术结构图

■ 在用户输入表示为文本后，Siri利用活跃本体和对应的执行系统将文本表示解析为用户意图。

■ 理解用户意图只是一个中间过程。为了能够帮助用户完成事务，Siri的服务系统将调用各种内部和外部服务，作为满足用户意图的结果。

■ 输出系统将Siri的运算结果返回给用户，完成一次服务，输出系统同样支持多模态输出方式。

Siri的资源主要分为数据资源和计算资源两类，其中数据资源有领域模型、词汇表数据库、短期记忆系统、长期记忆系统、领域本体数据库、对话流模型、服务模型、服务能力模型和外部服务；计算资源包括语音识别系统、语言模式识别器、语言解释器、对话流控制器、任务控制器、服务集成模块和语音生成系统。

Siri系统的复杂性由上面众多的数据资源和计算资源可窥一斑。下面将讲述输入系统和活跃本体两大子模块。

输入系统

Siri系统以用户输入开始，以输出服务结果为终，其间与用户进行多轮交互。Siri的输入系统主要提供了以下三个功能。

■ 支持多通道输入模式。所谓多通道，即不仅支持键盘文本输入模式，也支持语音自动识别模式、GUI界面操作模式以及系统事件触发的输入等多种输入模式。

■ 消除用户输入信息的歧义。自然语言理解系统的最大难点是在各级别的处理过程中会遇到形式各异的语言歧义现象，而输入系统也会对用户输入的信息进行初步的歧义消解工作。

■ 尽可能将用户的注意力引导到Siri能够提供的服务上。虽然Siri可用于趣味性的聊天对话，但其根本目的并不在此，能够为用户提供各个领域的实用服务是其本原，因此在Siri整个运作流程中都会尽量将用户引导到其能够提供的服务，而输入系统也肩负这一重任。

多模态输入

Siri提供多模态输入，即给用户多种可选择的输入方式（如图3所示）。用户可以选择文本键入方式，也可以选择语音输入方式或者在GUI界面通过点选来进行功能选择。

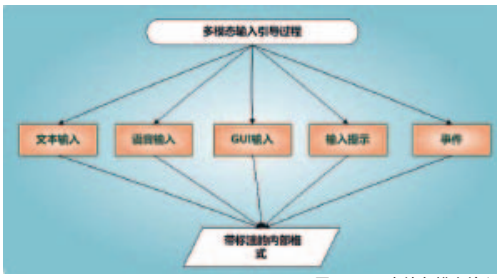


图3 Siri支持多模态输入

Siri提供了多种输入通道，供用户随意选择。在用户输入前，根据与用户个人兴趣的相关性提供给用户一些可能感兴趣的选择项。如果用户没有从提供的选择项里选择，则接受用户的主动输入，在用户输入过程中Siri自动判断用户输入是否完成。如果判断用户还需继续输入，则将当前的输入追加到之前的输入流中，同时根据与目前输入数据的相关性自动给出一些输入建议。如果判断用户输入完成，则结束输入过程。

在用户以任意一种输入方式输入信息的过程中，Siri都会实时地进行引导或提示，尽可能将用户的输入引导到Siri可以识别的商业实体或能够提供服务的功能上。归根结底，Siri是要根据用户的输入提供一些服务，采用主动引导而非被动等待用

户自由输入，能帮助用户节约时间，充分展示出Siri的能力。

Siri的输入提示是引导用户的重要组件。在用户输入部分信息后，Siri根据目前的信息，结合自己能够提供的服务，提供了功能强大的输入提示来引导用户到某项功能上来。这种输入提示可以是单词这种表层拼写级别的，也可以是语法语义和任务功能等深层级别的。

比如，最简单的输入提示是自动补全功能，用户输入了“rest”，此时Siri可以提示用户restaurants，通过直接选择减少用户的输入成本。Siri也提供如替换和语义改写类型的输入提示，比如用户输入：find restaurants of style，此时Siri可以提示用户：Italian，而用户点选了这个提示后，Siri可以将这个输入语义改写为：Find Italian restaurants。这是Siri在输入阶段就体现出来的智能。

语音输入流程

语音自动输入也是Siri最吸引人的特性之一。图4展示了Siri的语音输入流程。系统接收用户的声音流信号，通过调用“语音文字转换器”来将语音信号转换成可能的文字信号，这个“语音文字转换器”是Nuance通信公司提供的。

虽说如此，但这个转换器可被认为是Nuance通信公司为苹果公司特别制作的。我们知道，“语言模型”在语音识别系统中起到至关重要的作用，因为音频符号转换为文字信息时，有多种可能选项，而哪个选项才是用户真正表达的内容呢？此时语言模型在进行正确答案选择时会发挥很大的作用。

举个例子，假设用户语音输入：“我喜欢苹果”。语音识别后输出两个可能答案：1.我喜欢苹果；2.我稀饭苹果。此时可以利用语言模型来对两个结果进行打分，语言模型本质上是通过大规模语料统计来获得相连两个单词的连接紧密程度，通过语言模型可知“喜欢苹果”比“稀饭苹果”更容易在真实语料中出现，于是选项1的得分就高于选项2，此时Siri会将选项1作为识别结果。

之所以说这个转换器是为苹果公司定制的，是因

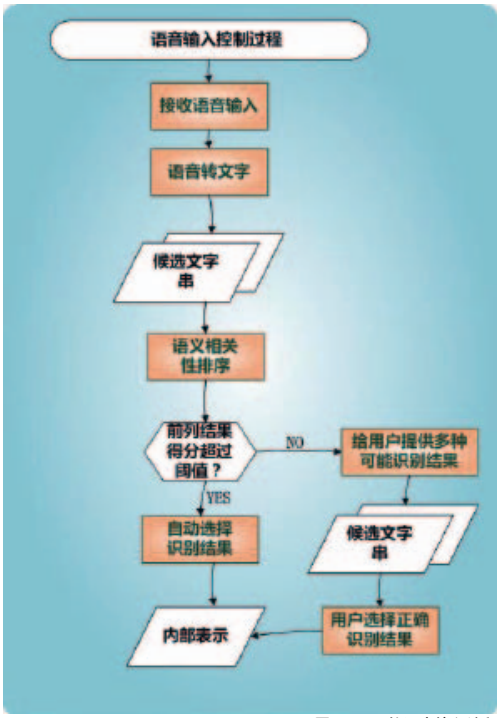


图4 Siri的语音输入流程

为转换器的语言模型是根据Siri专门训练的。Siri是多个垂直领域的垂直智能助手，所以数据中包含领域相关的商业实体名称以及用户的个性化信息。在训练语言模型时也会考虑这一点，并将这些信息在语言模型中的权值加大，这样在识别时更容易被识别为能由Siri提供的服务。所以说整个Siri系统都有引导用户到其能够提供的服务上来的机制，这点从语音识别中也可看出。

尽管经过语言模型调整有助于明确输入内容，但语音识别结果仍然不是唯一可确定的，会有语音流的多种文本解释。此时Siri会对其语法语义进行检查，判断哪些是符合语义的，哪些是不符合语义的。

比如语音识别结果为两个候选项：1.Italian food for lunch；2.Italian shoes for lunch。此时Siri可以判断出选项1在语义上更合理，所以会根据语义合理性进行排序。如果得分最高的选项超过一定阈值，则Siri认为这就是语音识别的结果，并进行一些语义标注工作作为最终识别结果。否则，会将多个选项提交给用户进行选择，把用户的选择结果作为最终识别结果。在上面的例子中，Siri可能会问用户：“Did you mean shoes or food?”然后根据用户的选择来挑选正确答案。

活跃本体

活跃本体和执行系统是Siri中最具技术含量的两个核心组件。活跃本体可以被理解为整个Siri系统的具体执行环境和场所；执行系统调用所有系统数据、词典、模型和程序，在活跃本体内将用户输入的信息解析为用户真正的意图，然后根据意图来调用外部的服务。

如果进行类比的话，可将活跃本体类比为人工智能系统里著名的“黑板系统”，即将数据放入黑板中，各个程序来对黑板内的数据进行加工。

在程序执行时，活跃本体内放入的数据和模型包括：领域模型、用户个性化信息、语言模式、词汇表和领域实体数据库等。

领域模型

领域模型包括某个领域内的概念、实体、关系、属性和实例的内部表示，这其实就是Semantic Web研究领域常说的ontology。Siri包含很多垂直领域的领域模型，如外出就餐领域包括餐馆这个概念，而这个概念又具有名称、地址和电话等属性；另外就餐事件（Meal Event）也是该领域一个概念，具有人数、日期和时间等属性；多个领域的领域模型被Siri用来引导输入、解析自然语言、派发服务和生成输出内容。

领域模型有以下一些功能。

- 表层单词到领域概念的映射关系。
- 单词分类，比如判断某个单词是个餐馆名称。
- 保存概念关系，比如“菜系”和“餐馆”的关系，以方便Siri在内部作出推理功能。假设用户输入是local mexican food，因为领域模型体现了菜系和餐馆其中一个是概念，另一个是概念的属性，于是通过内部推理可以表示为：find restaurants with style=mexican。
- 组织领域有关的服务地址，比如哪个网站提供餐馆评论。
- 在输出系统生成语言时，提供信息并进行语义转写。

领域模型采用被动触发机制，当外部任务或事件达到一定阈值后相关概念会被触发。

个性化信息

Siri在个性化方面也做得非常出色。在和用户沟通的过程中，一台本来冰冷无知的机器能够叫出你的名字，并且知晓你的个人爱好，用户体验无疑会非常好。

从具体技术手段上，Siri有长期记忆和短期记忆两个记忆系统，来实现个性化和与用户交流。短期记忆和长期记忆是存储个性化信息的场所，主要功能有两个：一方面在Siri将文本映射为语义或者用户意图时消除歧义，帮助Siri更准确地判断用户想要干什么；另一方面可以增加亲和力（知道用户的个人信息，在交互时增加亲切感）。

Siri的**短期记忆系统**会记录以下内容。

- 最近用户和Siri的完整对话记录。
- 最近用户通过GUI界面所做的选择项，比如打过哪些电话、浏览过哪些页面、查询过哪些商业实体查的地图等。
- 最近用户发出的查询请求及对应的查询结果。
- 最近用户和Siri交互时对应的传感器信息，比如地理位置、时间、手势等信息。

短期记忆系统是通过触发机制发挥作用的，比如当一些条件或者事件发生时，如果强度超过一定阈值，就会触发短期记忆系统。

长期记忆系统会记录以下内容。

- 用户个人信息，如名字、居住地址、账户信息及个人偏好等。
- 记载用户收集的相关信息，比如收藏书签等。
- 记录用户使用过的商业实体，比如电影院、餐馆、旅馆等，不仅记录其名称，还记录电话、地点等更详细的完备信息。
- 记录娱乐相关实体，比如电影、视频、歌曲等。
- 个人计划或者社交关系，比如待做列表、定时提醒或者其电话联系人列表等。
- 商品或者服务的历史购物记录或者心愿表。

与短期记忆系统类似，长期记忆系统也通过触发机制发挥作用。用户和Siri交互的任何时刻都有可能触发长期记忆系统，比如用户浏览网页或者在进行网络购物都可能触发长期记忆系统来获取其个人偏好信息。另外，长期记忆保存的账户信息也能避免用户输入繁琐的登录步骤。

语言模式识别系统

语言模式识别系统是对用户输入的表层、语法层、习惯用语和成语等进行模式匹配的模块。在Siri内部，这个系统的数据以模式定义的形式存在，而有专门代码解析这个模式定义数据。其实这就是我们平常说的模板系统。

匹配模式的代码在Siri内部采用正则表达式或者状态机等方式实现。这个模式识别系统的使用场合包括引导用户输入、NLP解析、识别任务类型和对话类型等。比如对于常用语what's happening来说，这就是一个具体的模式。在Siri匹配发现这个模式后，会在内部被认为是事件计划任务以及本地事件类型任务，这有助于进一步处理。

词汇表

词汇表用于维护Siri中的表层单词到领域模型或任务模型中定义的的概念、关系、属性的映射关系，被用来引导用户输入、自然语言解析和生成输出结果。

领域实体数据库

领域实体数据库是被Siri的执行系统直接使用的，其主要目的是存储领域实体的比较全面的信息并提供统一的查询接口。其存储的领域实体包含以下几方面。

- 商业类型实体，比如与餐馆相关的信息（名称、地址、电话等信息）。
- 娱乐类实体，如电影、视频、歌曲等实体信息。
- 其他领域的实体。
- 事件。
- 地理位置信息，比如著名地点、地标型建筑、机场及国家区划等地理信息。

假设用户的输入需求是要寻找一个合适的就餐地点，图5展示了Siri在某个运行时刻时活跃本体的即时快照。从图5中可以看出，在活跃本体中已经存储了被激活的相关概念及其属性，其中餐馆和就餐事件是两个具体的激活概念，而餐馆这一概念又包含名称、菜系、地理位置三个属性信息，地理位置本身既是一个属性，也是一个概念，而

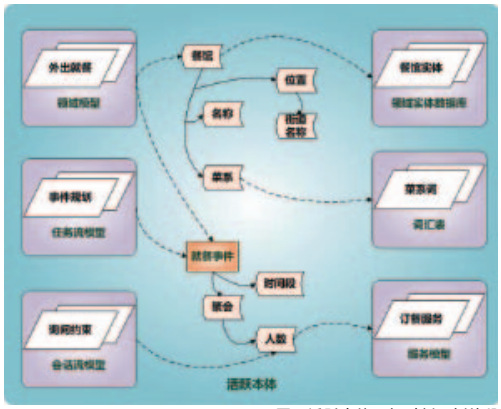


图5 活跃本体运行时刻示例片段

街道名称又是地理位置这一概念的属性。同样的就餐事件也有时间段、就餐人数等属性。在运行时刻，活跃本体中的餐馆概念可能已经根据领域实体数据库中的餐馆名称将某个具体餐馆具象化，即名称内已经填写入候选的具体餐馆名称作为这个激活概念的实例。

上述被激活的概念属性信息属于领域模型中的外出就餐这一具体垂直领域中的概念。而就餐事件同时也属于一种具体的任务流，隶属于任务流模型中的事件规划这一具体通用任务模型的领域子类型，用于确定任务的步骤和流程。假设此时Siri已经获得了一些用户输入信息，但还缺乏就餐人数这一约束条件，会话流模型会发现这一点并据此询问用户就餐人数信息。



张俊林

《这就是搜索引擎：核心技术详解》作者，新浪微博研发人员，中科院软件所博士，主要研究方向为自然语言处理、搜索技术、推荐系统及机器学习。

责任编辑：杨爽（yangshuang@csdn.net）

大话推荐系统评测

文 / 项亮

推荐系统作为一种信息过滤的重要手段，在解决信息超载问题上非常有潜力。而怎样评测推荐系统的好坏难之又难。本文中给出了多项评测推荐系统的指标，并介绍了其含义和获取方法。

完整的推荐系统一般有三个参与方：用户、物品提供者和提供推荐系统的网站。以图书推荐为例。首先，推荐系统要满足用户的需求，为用户推荐感兴趣的图书。其次，推荐系统要能将各出版社的书分别推荐给感兴趣的用户，而不是只推荐几个大出版社的书。最后，好的推荐系统要能够收集高质量的用户反馈，不断提高推荐的质量，增加用户和网站的交互，提高网站收入。因此在评测一个推荐算法时，要同时考虑三方利益。

为了全面评测推荐系统对三方利益的影响，本文从不同角度出发提出了推荐系统评测的多项指标，并依次介绍这些指标的含义及获取办法。

推荐系统实验方法

在介绍推荐系统评测指标之前，需要先了解计算和获得这些指标的三种主要实验方法：离线实验、用户调查和在线实验。下面将分别介绍这三种实验方法的优点和缺点。

离线实验

离线实验一般由以下几个步骤构成。

- 通过日志系统获得用户行为数据，并按照一定格式生成标准的数据集。
- 按照一定规则将数据集分成训练集和测试集。
- 在训练集上训练用户兴趣模型，在测试集上进行预测。
- 通过事先定义的离线指标评测算法在测试集上的预测结果。

可以看出，推荐系统的离线实验是在数据集上完成的。它的好处是不需要真实用户参与，可快速地计算出结果，能方便快速地测试大量算法。但它的主要缺点是无法获得商业上关注的一些指标，如点击率、转化率等，而且找到和商业指标非常相关的离线指标也是件很困难的事情。

用户调查

因为离线实验的指标和实际商业需要优化的指标之间存在差异，所以要准确评测一个算法，就需要相对比较真实的环境。最好的方法是将算法直接上线测试，但如果测试具有较高风险，则需要在上线测试前进行一次被称为用户调查的测试。

招聘一些真实用户，在需要测试的推荐系统上完成一些任务。在用户完成任务时，观察和记录他们的行为，并让他们回答一些问题。之后，通过分析用户行为和答案来了解测试系统的性能。

用户调查的优缺点很明显。它的优点是可以获得很多用户主观感受的指标，与在线实验相比，风险很低，出现错误后很容易弥补；缺点是招聘测试用户代价较大，很难组织大规模的测试，从而造成测试结果的统计意义不足。此外，在很多时候设计双盲实验非常困难，而且用户在测试环境下的行为和真实环境下的行为可能有所不同，这就会造成测试环境下收集的测试指标在真实环境下无法重现。

在线实验

在完成离线实验和必要的用户调查后，可将推荐系统上线做AB测试，来将它和旧算法进行比较。

AB测试（AB Test）是一种很常用的在线评测算法的实验方法。它通过一定规则将用户随机分成几组，不同组的用户用不同算法，然后统计不同组用户的各种评测指标来比较算法。比如可以通过点击率比较算法的性能。

AB测试的优点是可以公平地获得不同算法实际在线时的性能指标，其中包括商业上关注的指标；缺点主要是周期比较长，必须经过长期的实验来得到可靠的结果，因此一般不会用AB测试测试所有算法，而是用它测试那些在离线实验和用户调查中表现很好的算法。

一般来说，新的推荐算法最终上线前，需要完成上面所说的三种实验：

- 通过离线实验证明它在很多离线指标上优于现有的算法；
- 通过用户调查确定它的用户满意度不会差于现有的算法；
- 通过在线的AB测试确定它在我们关心的指标上优于现有的算法。

评测指标

下面将详细讨论各推荐系统评测指标。

用户满意度

用户是推荐系统的重要参与者，用户满意度是推荐系统需要满足的最重要指标。但用户满意度没有办法离线计算，只能通过用户调查或在线实验获得。

用户满意度主要通过问卷形式调查。可将用户对推荐系统的满意度分为不同层次。下面是一个论文推荐系统用户满意度的调查问卷。

- 推荐的论文都是我非常想看的。
- 推荐的论文很多我都看过了，确实是符合我兴趣的不错的论文。
- 推荐出来的论文与我的研究兴趣相关，但我并不喜欢。
- 不知道为什么会推荐这些论文，和我的兴趣丝毫没有关系。

这些问题从不同侧面询问了用户对推荐结果的感受。只有在设计问卷时全面考虑用户的感受，才能得到用户的准确回答。

有些网站会设计一些收集用户满意度的界面。比如在视频网站Hulu的推荐页面中就有满意或者不满意的反馈按钮让用户评价，通过统计两种按钮的点击情况就可以度量系统的用户满意度（如图1所示）。还可以用点击率、用户停留时间和转化率等指标来度量用户满意度。

预测准确度

离线的推荐算法有很多研究问题，这里将讲述评分预测问题和Top-N推荐问题，并针对不同的研究问题介绍预测准确度指标。

评分预测问题。很多提供推荐服务的网站都有一款让用户给物品打分的产品。知道了用户对物品的历史评分，就可以从中学习到用户的兴趣模

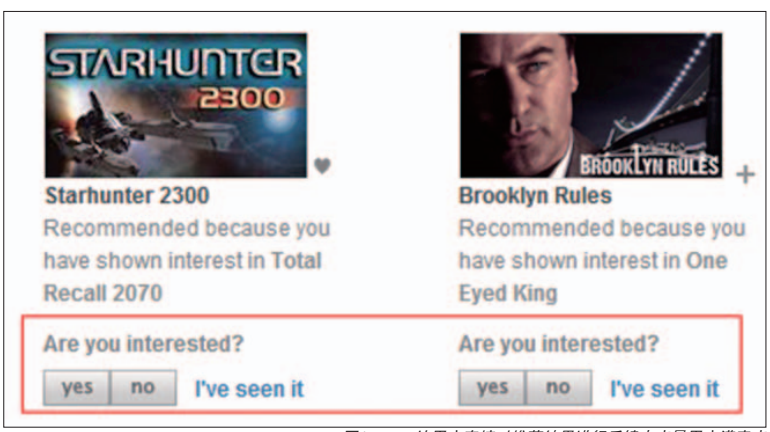


图1 Hulu让用户直接对推荐结果进行反馈来度量用户满意度

型，来预测一个用户在将来看到一个他没有评过分的物品时，会给该物品评多少分。预测用户对物品评分的问题，被称为评分预测问题。

评分预测问题的预测准确度一般通过均方误差（RMSE）和绝对误差（MAE）计算。对于测试集中的一个用户u和物品i，令r_{ui}是用户u对物品i的实际评分，而$\hat{r}_{ui}$是推荐算法给出的预测评分，那么RMSE定义为：

$$RMSE = \frac{\sqrt{\sum_{u,i\in\mathcal{T}}(r_{ui}-\hat{r}_{ui})^2}}{|\mathcal{T}|}$$

MAE采用绝对值来计算预测误差，它定义为：

$$MAE = \frac{\sum_{u,i\in\mathcal{T}}|r_{ui}-\hat{r}_{ui}|}{|\mathcal{T}|}$$

Top-N推荐问题。网站在提供推荐服务时，一般是给用户一个个性化的推荐列表，这种推荐叫做Top-N推荐。Top-N推荐的预测准确率一般通过准确率（Precision）/召回率（Recall）来度量。

令R(u)是根据用户在训练集上的行为给用户做出的推荐列表，而T(u)是用户在测试集上的行为列表。那么，推荐结果的召回率定义为：

$$\text{Recall} = \frac{\sum_{u\in U}|R(u)\cap T(u)|}{\sum_{u\in U}|T(u)|}$$

推荐结果的准确率定义为：

$$\text{Precision} = \frac{\sum_{u\in U}|R(u)\cap T(u)|}{\sum_{u\in U}|R(u)|}$$

有时，为了全面评测Top-N推荐的准确率和召回率，一般会先选取不同的推荐列表长度N，计算出一组准确率/召回率，然后画出准确率/召回率曲线（Precision/Recall Curve）。

覆盖度

覆盖度（Coverage）描述了推荐系统对物品长尾的发掘能力。覆盖度有多种定义方法，最简单的是，将其定义为推荐系统能够推荐出来的物品占总物品集合的比例。假设用户集合为U，推荐系统给每个用户推荐的物品列表为R(u)。那么覆盖度定义为：

$$\text{Coverage} = \frac{|\cup_{u\in U}R(u)|}{|I|}$$

覆盖度为100%的推荐系统可将每个物品都推荐给至少一个用户。可以看到，热门排行榜的推荐覆盖度很低，它只会推荐热门物品，这些物品在总物品中占的比例很少。

但上面的定义过于粗略。覆盖度为100%的系统可以有无数的物品流行度分布。为了更细致地描述推荐系统发掘长尾的能力，需要统计推荐列表中不同物品的出现次数的分布。如果所有的物品都出现在推荐列表中，且出现的次数差不多，那么推荐系统发掘长尾的能力就很好。在信息论和经济学中有两个著名的指标可以用来定义覆盖度。第一个指标是信息熵：

$$H = -\sum_{i=1}^{n} p(i)\log p(i)$$

第二种指标是基尼系数（Gini Index）：

$$G = \frac{1}{n-1}\sum_{j=1}^{n}(2j-n-1)p(i_j)$$

这里，$p(i_j)$是按照物品热门度p()从小到大排序的物品列表中第j个物品。

多样性

用户的兴趣一般比较广泛。在一个视频网站中，用户可能既喜欢看《猫和老鼠》一类的动画片，也喜欢看成龙的动作片。那么，为了满足用户广泛的兴趣，推荐列表要尽量覆盖用户的所有兴趣点，即推荐结果要具有多样性。

尽管用户的兴趣在长时间内是多样的，但具体到用户访问推荐系统的这一刻，用户的兴趣往往是单一的。如果推荐列表只能覆盖用户的一个兴趣点，而恰好不是用户这个时刻的兴趣点，用户就会不满意。

多样性描述了推荐列表中物品两两之间的不相似性。因此，多样性和相似性是对应的。假设定义了物品i和j之间的相似度，那么用户u的推荐列表R(u)的多样性定义如下：

$$\text{Diversity} = 1 - \frac{\sum_{i,j\in R(u),i\neq j}s(i,j)}{\frac{1}{2}|R(u)|(|R(u)|-1)}$$

而推荐系统的整体多样性可以定义为所有用户推荐列表多样性的平均值：

$$\text{Diversity} = \frac{1}{|U|}\sum_{u\in U}\text{Diversity}(R(u))$$

从上面的定义可以看出，不同的物品相似度度量函数s(i,j)可以定义出不同的多样性。如果用内容相似度描述物品间的相似度，就可以得到内容多样性函数；如果用协同过滤的相似度函数，就可以得到协同过滤的多样性函数。

新颖性

新颖的推荐是指给用户推荐那些他们以前没有听说过的物品。在一个网站中实现新颖性的最简单的办法是把那些用户之前在网站中有过行为的物品从推荐列表中过滤掉。比如在一个视频网站中，新颖的推荐不应该给用户推荐那些他们已经看过、打过分或浏览过的视频。而有些视频可能用户在别的网站看过，或者是在电视上看过。因此，只过滤掉本网站中用户有过行为的物品还不能完全实现新颖性。

最简单的评测新颖性的方法是利用推荐结果的平均热门度。因为越不热门的物品用户觉得新颖的可能性越大，所以如果推荐结果中物品的平均热门程度较低，那么推荐结果的新颖性越高。但用推荐结果的平均热门度来度量新颖性比较粗略，因为不同的用户不知道的东西不同。因此，要准确地统计新颖性需要做用户调查。

最近几年关于多样性和新颖性的研究越来越受到推荐系统研究人员的关注。ACM的推荐系统会议在2011年有一个讨论会专门讨论推荐的多样性和新颖性。该研讨会的组织者认为，牺牲精度来提高多样性和新颖性很容易，而困难的是如何不牺牲精度来提高多样性和新颖性。

惊喜度

惊喜度（Serendipity）是最近这几年推荐系统领域最热门的话题。目前并没有什么公认的惊喜度指标定义方式，这里只给出一种定性的度量方式。上面提到，令用户惊喜的推荐结果是和用户历史上喜欢的物品不相似，但用户却觉得满意的推荐。那么，定义惊喜度需要先定义推荐结果和用户历史上喜欢的物品的相似度，然后要定义用户对推荐结果的满意度。

前面提到，用户满意度只能通过问卷调查或者在线实验获得。而推荐结果和用户历史上喜欢的物品相似度一般可以用内容相似度定义。也就是说，如果获得了一个用户观看电影的历史，得到这些电影的演员和导演集合A，然后给用户推荐一个不属于集合A的导演和演员创作的电影，而用户表示非常满意，这样就实现了一个惊喜度很高的推荐。因此提高推荐惊喜度需要在提高推荐结果的用户满意度的同时降低推荐结果和用户历史兴趣的相似度。

在电子商务推荐系统中，用户对推荐结果的信任度非常重要。以让用户信任的方式推荐更能引发用户去购买。而类似广告形式的推荐，则很难让用户产生购买意愿。

信任度

如果你有两个朋友，一个人你很信任，另一个人经常满嘴跑火车，那么你信任的朋友推荐你去某个地方旅游，你很有可能会听从，但那位满嘴跑火车的朋友推荐你去同样的地方旅游，你很有可能不去。可以将这两个人看做两个推荐系统，尽管他们的推荐结果相同，但用户却产生了不同的反应，这就是因为用户对它们有不同的信任度。

对于基于机器学习的自动推荐系统，同样存在信任度（Trust）的问题。如果用户信任推荐系统，那么与推荐系统的交互就会增多。特别是在电子商务推荐系统中，让用户对推荐结果产生信任非常重要。同样的推荐结果，以让用户信任的方式推荐给用户就更能引发用户去购买，而以类似广告形式的方法推荐给用户，就很难让用户产生购买意愿。

度量推荐系统的信任度只能通过问卷调查的方式，询问用户是否信任推荐系统的推荐结果。

提高推荐系统信任度的方法主要有如下两种。

- 需要增加推荐系统的透明度（Transparency）。而增加推荐系统透明度的主要办法是提供推荐解释。只有让用户了解并认同推荐系统的运行机制，才会提高用户对推荐系统的信任度。
- 要考虑用户的社会网络信息，利用用户的好友信息给用户推荐，并通过好友解释推荐。这是因为用户对好友一般都比较信任，所以如果推荐的

商品是他们的好友购买过的，那么他们对推荐结果就会相对比较信任。

实时性

因为有些物品（新闻、微博等）具有很强的时效性，所以需要及时将它们推荐给用户。比如，给用户推荐旧闻的价值显然不如推荐新闻。这时，推荐系统的实时性至关重要。

推荐系统的实时性包括以下两个方面。

- 推荐系统需要实时更新推荐列表来满足用户新的行为变化。很多推荐系统每天离线计算一次用户推荐列表，然后在线将推荐列表展示给用户。这种设计显然无法满足实时性。与用户行为相应的实时性，可以通过推荐列表的变化速率来评测。如果推荐列表在用户有行为后变化不大，或者没有变化，说明推荐系统的实时性不高。
- 推荐系统需要能够将新加入系统的物品推荐给用户。这主要考验了推荐系统处理物品冷启动的能力。如何将新加入系统的物品推荐给用户是个复杂的问题，这里将不会详细讲述。而新物品推荐能力的评测指标，可以利用用户推荐列表中有多大比例的物品是当天新加的物品来评测。

鲁棒性

任何一个能带来利益的算法系统都会被人攻击。最典型的例子是搜索引擎。目前，推荐系统也遇到了作弊问题，而鲁棒性指标衡量了一个推荐系统抗击作弊的能力。

在2011年的推荐系统大会上，专门有一个关于推荐系统鲁棒性的教程【注：Neil Hurley. Tutorial on Robustness of Recommender System. ACM RecSys 2011】。其中总结了很多作弊的方法，最著名的就是行为注入攻击（Profile Injection Attach）。

众所周知，绝大部分推荐系统都是通过分析用户的行为来实现推荐算法的。比如，Amazon有一种推荐叫做“购买商品A的用户也经常购买的其他商品”。它的主要计算方法是统计购买商品A的用户购买其他商品的次数。那么，可以注册很多账号，用这些账号同时购买A和自己的商品，从而提高自己商品的排名。还有一种攻击主要针对评分系统，比如雇用一批人给自己的商品评高分。

算法的鲁棒性主要通过模拟攻击来评测。首先，给定一个数据集和一个算法，可以用这个算法为这个数据集中的用户生成推荐列表。然后，用常用的攻击方法向数据集中注入噪声数据，再利用算法在注入噪声后的数据集上给用户生成推荐列表。最后，通过比较攻击前后推荐列表的相似度来评测算法的鲁棒性。如果攻击后的推荐列表相对于攻击前没有发生大的变化，就说明算法比较鲁棒。

小结

本部分内容提到了很多指标，如表1所示。如何优化离线指标来提高在线指标是推荐系统研究的重要问题。这个问题目前仍然没有什么定论，只是不同系统的研究人员自己有一定的感性认识。

表1　获取各种评测指标的途径

	离线实验	问卷调查	在线实验
用户满意度	×	√	○
预测准确度	√	√	×
覆盖度	√	√	√
多样性	○	√	○
新颖性	○	√	○
惊喜度	×	√	×

一般认为应该在给定覆盖度、多样性、新颖性的限制下，尽量优化预测准确度。用一个数学公式表达，离线实验的优化目标是：

$$\max \quad \text{预测准确度}$$
$$\text{s.t.} \quad \text{覆盖度} > A$$
$$\text{多样性} > B$$
$$\text{新颖性} > C$$

其中，A、B、C的取值应该视不同的应用而定。

本文节选自人民邮电出版社即将出版的《推荐系统实践》一书。本书详细介绍了推荐系统的各方面技术，介绍了实际推荐系统中常遇到的问题以及解决方案。特此感谢人民邮电出版社图灵公司授权。本文选登时有删减。

Hypertable应用实践
基于BigTable模型优化业务

文 / 杨栋

在2月刊《程序员》刊登的《Hypertable应用实践：比肩HBase》一文中，作者主要从Hypertable的系统层面介绍了如何改进和优化系统来更好地支持业务。本文则主要从业务层面介绍如何基于BigTable模型设计表格及导入/查询策略。

NoSQL和SQL数据库谁更重要的争论从来都没停止，甚至NoSQL中哪些软件更好的争论也很多。和大多数人的观点类似，我的看法是基础架构软件的使用取决于其应用场景和系统成熟度，也取决于研发/支撑团队的能力。

在互联网平台软件行业里，有一种论调“大者恒大”。那些大家普遍看好的平台软件，会有更多人聚在其周围，会有更多人贡献。从长远来看，使用这种软件可以得到更好的生态系统支持。

HBase和Hypertable是BigTable的两个开源产物。HBase的社区影响力比Hypertable大，使用HBase的公司也更多。不过在这里我却持相反的论调，因为所有使用开源软件的公司，内部都会对其进行一些定制和优化，内部维护的版本和社区版本有一定差别。

就社区版本而言，由于为HBase贡献代码的人数众多，所以引入的Bug也相应会多；而Hypertable的主体开发主要由少数核心人员完成，并且目前Doug Judd成立了专门的公司来维护，因此在稳定性上不比HBase差，性能上又超过HBase。这对于强调节约成本的互联网公司来说，是个不小的诱惑。

社区论坛

在社区论坛中，不同用户可以在不同讨论区中的不同主题下跟帖。假设考虑使用Hypertable作为论坛数据的底层存储。

建表策略

主表的字段划分。在实现系统中，同一个帖子的标题、内容等数据存放在同一个字段（Column）中，而不是像传统数据库那样分别存放在不同的Column中，主要出于以下考虑。

- 由于Hypertable的数据存储是key-value方式，因此每个字段都要存放一份Row key和Column name等，若字段数很多，存储开销较大。
- 除删除操作外，基本不存在单独修改一个字段的情况。而删除只需要打一个管理标记，如果增加一个单独的字段存放管理标记（flag字段），则可以完美支持删除操作。
- 几乎所有的查询操作都是查询整个帖子的内容，即使存在只查询部分字段的需求，也可以通过客户端过滤实现，所损失的仅仅是部分带宽。

索引表的主键设计。Hypertable只支持主键（row key）索引，因此要加速根据其他字段内容进行的查询，只能通过创建索引表的方式。本系统中所有的索引表均采用index value:date为Row key（例如userid:date），这样设计主要出于以下考虑。

- 既可以支持单纯按index value查询，如查询user id = 100发的所有帖子，只需指定“100:”< row <“100;”；也可以附加时间段限制，如查询user id = 100在北京时间2009年1月1日发的所有帖子：“100:1230796800” <= row <=“100:1230883200”。
- 防止一个index value下的帖子过多导致溢出。BigTable有一个限制，确保同一个Row key下的数据不能超过某个阈值（例如200MB），而增加时间后缀可以有效规避这个限制。
- 对于简单的复合查询，如某个用户在某个讨论区中发的所有帖子，也可以通过简单的index value叠加建立索引表。但对于较复杂的查询，如某一组IP在某时间段内发在某几个讨论区中尚未被删除的帖子，使用简单的索引表就无能为力了。这种情况只能通过客户端逻辑对查询条件进行拆分，对查询结果进行过滤、求交集等操作来实现，很难做到实时。好在底层系统可以不考虑这类查询需求，交给上层Cache系统即可。

为了支持高并发，Hypertable采用MVCC模式存储key和value，即删除一个value只是追加了一个补丁，而不是在原值基础上修改。

频繁读-删除-更新模式。在最初的表格设计中，为了在BigTable现有功能框架上模拟按顶帖时间排序的功能，使用了一个ThreadByReplyTimeIndex表（key为forumid:最后回复时间按位取反:threadid，value为threadid）和一个ThreadInfo表（key为threadid，内容包括帖子数和最后回复时间等）。这样每次写入一个新帖子，就必须从ThreadInfo表读取本thread上一个post的回复时间，删除ThreadByReplyTimeIndex表中对应的那一行，然后重新插入一行。

要实现这种逻辑，必须每写一个帖子就flush一次，并且增加很多随机读请求。经测试，采用这种模式最初的写入速率很低，而且随着数据量增加，一次随机读的响应时间也随之增加，写入速率会很快下降。因此，如果需要对表格数据进行更新，或者涉及随机读，就要将该表格设置为内存表，提高访问效率。

但in memory模式非常耗费内存，原因如下。

- 为了支持稀疏表，Hypertable每个value是单独存的，而不是按行存，因此每个value都需要存一份key（包括Row key、Column family、Column qualifier和Timestamp，最小开销16字节），再加上map数据结构的开销24字节，一个value至少有40字节额外开销，一个帖子就是40×13=520字节，比帖子的实际内容（平均300多字节）还多。
- 为了支持高并发，Hypertable采用MVCC（multi-version concurrency control）模式存储key和value，即删改一个value只是追加了一个补丁，而不是在原值基础上修改。只有当Cell Cache大小达到一定程度时才会清理多余的版本。

系统优化

数据更新压力。数据更新模式可能会对集群系统造成较大压力，假设BigTable的RangeServer每隔10秒会刷写一次CellCache，而线上流量实时更新时，BigTable每秒接收的数据量不会很大。

如果1秒接收350个命令包，平均每个消息大小约500字节（包含删除命令），那么每秒接收不到200KB的数据，10秒接收不到2MB的数据，造成了HDFS上写入很多小文件，再频繁地合并、删除这些小文件，则对HDFS造成较大压力。

如果刷写CellCache的间隔是5分钟，那么5分钟接收的60MB数据将会一次写到磁盘，再做合并、删除等操作的次数将会大大减少，这样HDFS的压力就会小得多。

数据导入压力。考虑多个进程并发地向集群导入历史数据，每个进程读取部分原始数据文件，实际执行中发现某个RangeServer的压力会增大，内存占用会在短时间内骤然增大，甚至出现内存耗尽的现象。

这里涉及BigTable系统两个方面的并行，一是Range间的并行，即多个导入的数据文件分别对应不同的Range；二是Range内的并行，即一个导入文件跨越多个Range。

BigTable模型中Range分裂时前半部分会被移走，该策略使得导入多个数据文件时，某台RangeServer总会接收各数据文件的流量，压力过大。为了防止压力集中，导入数据时原始文件按照PostID（Rowkey）的倒序排列，这使得每当Range分裂到其他RangeServer时，数据流量也会转移。

另外，社区的主表（POST）只有一个，Rowkey值是PostID。原始文件中每条记录的PostID都是顺序排列的，那么将原始文件的数据导入集群时，Range的负载会落到一个RangeServer上。

历史超链

需求描述

链接可以说是互联网最重要的“推荐”（或投票）形式，而推荐本质上是人的一种观念，是与时间相关的。将超链的产生和消亡历史存储下来，构建一个历史超链信息库，可以用于数据挖掘和分析，例如过期域名过滤、超链反作弊分析和用户等级区分等。

简单来看，超链状态有三种：新链接（new）、删除链接（del）和Anchor变化链接（mod）。状态转化如图1所示。

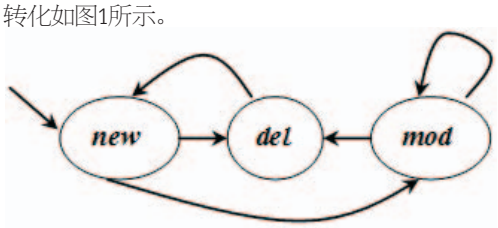


图1　历史超链信息库记录状态转化示意图

超链历史存储模块每天例行运行，超链变化信息归并到历史数据库。读取库数据时，可能只访问最新版本，也可能访问所有历史版本记录。比如统计记录状态经常变化到del的频度，可以发现死链。总的来说，可以将历史库的应用需求列为如下三点。

1. 需要统计出链（To）的变化，可能要统计所有出链，而不只是一个入链（From）下的。

2. 需要统计最新的To。

3. 如果From和To是分开存储的，最好把#号后面的那些字段（Anchor）和From一起存储，因为这些字段也是有用的。

对于需求2，可将From作为key，该页面将From对应的多个To作为value，每次更新数据时以From为单位。Hypertable支持多版本存储，在扫描时指定max_versions为1就能获得最新版本的To。

对于需求3，同样可以将From作为key，To为Column，其他字段作为1个或多个Column。

对于需求1，现有系统在历史库中保留了每条记录（To）的状态，可扫描记录的所有版本，获得链接的变化（状态）序列。新库中每条记录的导入都需要在历史库中做若干次查询，这种随机查询的性能很低（几毫秒到几百毫秒），直接影响到新库记录的导入速率。历史超链信息库的初始数据量超过数百TB，每天产生几TB的新库量。

假设每条记录大小为0.1KB（已结构化），即使每天24小时不间断插入，每秒也需要近几十万次的插入速率，即每次插入前的查询需要小于2微秒，这显然做不到。

为了减少或避免插入记录时的查询动作，我们需要找到另一种方法来表示页面（From）链接（To）的变化序列。

建表策略

方案一

从图1来看，链接的状态变化是无序的，因此我们不能通过有序的序列来表示它，最直接的还是通过记录本身的存储方式来表示，以便使用超链库的应用程序能够通过记录本身的存储方式转化（间接）得到链接的变化序列。

一个方案是稀疏表，如图2所示，可以将From作为表的Rowkey，将多个To作为多个Column。假设追加导入3次后，形成图2中的表格，那么使用超链库的应用可以通过扫描From行Ton列的所有版本得到Ton链接的变化序列，即new→mod；同

Row Key	Column Value			
From / To	weibo.com/csdn	csdn.net	programmer.com.cn/7552/	cloud.csdn.net/a/20120315/313137.html
www.programmer.com.cn/9642/	Anchor			
		Anchor	?	Anchor
			Anchor	

图2 历史超链库稀疏表示意图1

理，To1为new；To2为new→delete；Tok为new→delete。

方案二

图2示意的设计方式可以满足需求3中对于链接历史状态变化序列的需求。不过这种方式存在以下一些新的问题。

- 与table Schema相关的额外操作。每次插入时需要去查询Schema。如果是新链接插入，还需要修改Schema。
- 链接状态变化的表示。图2只示意了通过稀疏表的空闲位置和有效位置来共同表示链接状态的变化，但Hypertable插入下条记录时：

csdn.net$cloud.csdn.net/a/20120315/313137.html

不会在空白列自动添加字段，它的存储方式是Rowkey+Column+value。

针对这些问题，有以下几种解决办法。

- 在每次插入时，在相应的空列处插入特殊标记$。这种做法的弊端是增大额外存储（Rowkey+Column+$），把稀疏表变成了稠密表，因为超链库的记录是大key。
- 不在稀疏表的空白处做标记。在应用使用超链库时，扫描列的同时扫描行，再做交（Join）运算，这样能够发现相应的“空格”位置。但这种方式使得只扫描一列的开销变得很大。
- 单独设置一个静态值，保存插入超链历史库的记录数。每次插入时，将这个记录数也写到表格中。假设写到每一列中，结果如图3所示。

Row Key	Column Family			
From / To	weibo.com/csdn	csdn.net	programmer.com.cn/7552/	cloud.csdn.net/a/20120315/313137.html
www.programmer.com.cn/9642/	Anchor 3			Anchor 3
		Anchor 2		Anchor 2
			Anchor 1	
				Anchor 0

图3 历史超链库稀疏表示意图2

Row Key	Column Family				
From / To	weibo.com/csdn	csdn.net	programmer.com.cn/7552/	cloud.csdn.net/a/20120315/313137.html	TS
www.programmer.com.cn/9642/	Anchor				Day4
	Anchor	Anchor		Anchor Day3	Day3
			Anchor	Anchor Day1	Day1
		Anchor			Day0

图4 历史超链库稀疏表示意图3

通过将版本信息（记录计数）写入表格，使用超链库的应用能够读出版本序列。比如读cloud.csdn.net/a/20120315/313137.html，会得到版本变化序列0、2、3，它就表示了链接的状态变化，不连续的版本数表示了del和new的变化，连续的版本数表示了mod的变化。

不过这种方案的不足之处在于，插入计数是针对每个页面（From）的，即插入程序需要维护每个页面的计数，量级可能是百亿或千亿，每次导入都要先查询页面的计数。这些信息的存储将超过TB级，每次导入时这次额外的查询将比较费时。

方案三

为了避免这次计数查询，还需要一些额外信息，可从Hypertable的应用方面挖掘一些有效信息。已知每个页面的记录（From→To）每天有一条或者没有，那么我们可以利用时间戳。为表格新增一列，用来记录时间戳，通过其获得页面链接的状态变化序列。

应用程序在使用超链信息库时，需要扫描两列，一个是cloud.csdn.net/a/20120315/313137.html列（数据列），一个是TS列（时间列），如图4所示。

(value Day3)、(value Day1)是数据列，Day4、Day3、Day1、Day0是时间列。假设Day2那天没有这个页面的相关链接记录插入。首先将数据列和时间列按照时间一一对应起来，即Pair3<(null Day4), Day4>，Pair2<(value Day3), Day3>，Pair1<(value Day1), Day1>和Pair0<(null Day0), Day0>。

由于Pair0和Pair1时间按天连续，并且Ton列的value值从null→value，因此表示该链接（ton）的状态变为null→new。由于Pair1和Pair2时间

按天不连续（Day2未导入该页面相关的链接信息），对这种情况，依照按天连续处理，并且此时cloud.csdn.net/a/20120315/313137.html列的值从value→value，表示页面链接在这三天间的变化是new→mod。Pair2和Pair3时间按天连续，并且cloud.csdn.net/a/20120315/313137.html列的值从value→null，表示页面链接的变化是mod→del。

通过这个额外附加的时间列，我们能够在应用超链库时较方便地获取页面的状态变化，付出的代价是为超链表增加了1个稠密列。

表格的数目（是否所有记录存储一张表）是个问题，一种方案是整个库一张大表，From作为表的Row key，To作为表的各个列。假设超链库中每个页面大概有100个To链接，如果页面超过百亿，那么表格需要创建百亿（10^{10}）量级的列，虽然在逻辑上Hypertable是支持的。

另一种方案是每个页面（From）对应一个表，这样每张表的列数不超过100，但表格数目会超过百亿。从Root Range推算，假设Range阈值是1GB，每条Range的记录按照100字节计算，它能记录10^7个Meta Range，每个表至少一个Meta Range，即只能表示10^7个表格，距离10^{10}差距较大，因此第二种方案不适用。

当然，折中考虑，可以先将From（Row key）做哈希计算，再划分到各个表格，从而能够减少表格的列数。

假设将From集合划分成1万多个子集，每个子集对应一张表格，以From为key，每个To占一列。即便是这样划分之后，每个链接库表格的列数也会是10^6量级，这需要实际测试。

方案四

到此为止，我们通过方案三的设计能够满足超链库逻辑方面的需求，但系统实现支持这种设计是个很大的挑战。

超链库表格Schema使用From作为Rowkey、To作为Column，这个设计的优势在于超链库应用程序的易用性，直接扫描所需的列即能获取相关页面链接的信息；其劣势在于超链库的文件数可能达到很大，尤其是Column和AccessGroup的对应方式，这有可能导致超链库产生上亿的文件。就目前所知，支持这个量级文件数的Hadoop集群至少需要数百个节点。

Row Key	Column Family	
From:To	Anchor	Timestamp
www.programmer.com.cn/9642/:weibo.com/csdn	V'	Day4
	V	Day3
www.programmer.com.cn/9642/:csdn.net	V'	Day3
	V	Day0
www.programmer.com.cn/9642/:programmer.com.cn/7552/	V	Day1
www.programmer.com.cn/9642/:cloud.csdn.net/a/20120315/313137.html	V'	Day3
	V	Day1

图5 历史超链库稀疏表示意图4

为了满足超链库的可用性，可以放弃易用性，即From作Rowkey的建表方案，选择From:To作为Rowkey，如图5所示。

超链库的Column Family包括两列，Anchor和Timestamp，Timestamp记录着该条记录的时间。列数的固定（小于10个）使得整个表产生的文件数目大大减少，也无须在插入记录时考虑表的Schema。

选择这种建表方案，超链库的易用性降低了。使用超链库的应用想要获取某个页面（From）某个链接（To）的状态变化，需要首先将某个页面（From）的所有Timestamp列扫描出来，然后进行排序，比如图5会排序成Day0、Day1、Day3、Day4的时间序列，然后又可以效仿方案三中的策略，将时间序列和数据列做Join，获得指定From和To的链接的状态变化。

小结

总之，BigTable的模型和传统RDBMS有所不同，基于BigTable进行开发，除了考虑规模问题，还需要注意随机读性能会影响业务逻辑。如果需要基于BigTable的复杂操作，最好在Hypertable之上的内存中进行处理，再将计算结果导入系统。



杨栋

百度分布式高级研发工程师，从事Hypertable、Hadoop及流式计算的研究和开发。

责任编辑：杨爽（yangshuang@csdn.net）

云计算基础设施解析

文 / 郭理靖

借力云平台，很多公司在业务上取得了成功，但由于云平台上开发经验的欠缺，开发者在云平台的使用上存在不少误区。针对这一现状，本文详细讲述了使用EC2、EBS和S3三个云平台产品过程的常见问题，以及如何在云平台上架构稳定可用的服务。

在国外，基于Amazon AWS平台的成功案例越来越多，如日中天的Dropbox、提供DVD租借服务的老牌厂家Netflix，还有无数使用AWS的创业公司，都已证明AWS不仅是一个成功的商业模式，还是一个可以改变互联网传统开发模式的产品。

国内也有不少团队在使用Amazon AWS为他们的全球客户提供服务，但由于AWS访问速度的问题，很多开发者开始使用国内的云服务提供商的产品。可能是因为大家刚刚从传统的开发模式转移到云平台上，而且云平台上开发经验的分享相对较少，所以开发者在使用云平台上有不少误区。本文将详细介绍使用几个云平台（IaaS平台）产品过程中经常碰到的一些问题，以及如何在云平台上架构稳定可用的服务。

EC2/云主机

不管是从市场调查，还是从实际运营的数据来看，EC2/云主机是使用最多的产品，其概念也是用户最容易理解的。从技术角度来看，EC2/云主机背后的技术是虚拟化，而虚拟化技术是大家耳熟能详的，并且多多少少在工作过程中都接触过。虚拟化技术所带来的好处也广为人知，这里就不重述了。无论用户选择哪家云计算平台，只要申请了EC2/云主机，都可以像在使用一台真正的服务器一样，安装软件，部署程序，对外提供服务。

EC2/云主机最大的特点是按需申请、按需计费。用户可以根据自身需求，申请只有一个CPU且内存很小的超微型机器，也可以申请有多个CPU和大内存的超大型机器，这大大减少了创业团队的初期投入。因为按以前的开发模式，产品想上线，一般需要购买机器放到机房托管。而对产品上线后的流量估计又很难做到准确，机器买少了支撑不住整个产品，买多了又浪费。至于其他优点，如多种系统支持、方便的控制台管理等，其实都是一个好的云平台应该做的事。

经常有用户会问EC2/云主机可靠吗？性能好吗？使用EC2/云主机时应该注意什么问题？怎样在EC2/云主机上搭建高可用的服务？下面将针对这些问题一一作出解答。

EC2/云主机可靠吗

EC2/云主机是寄宿在物理机上的虚拟机，和物理机一样不可靠。一旦物理机出问题，就一定会影响到虚拟机，因此不要指望云主机像钻石一般，能够恒久远，永不宕机。所有物理机会出的毛病，EC2/云主机一样都会出。

EC2/云主机性能怎样

性能的具体指标很难定义，但可以从CPU、内存和磁盘I/O上粗略看出一台机器的性能好坏。先从CPU角度来说，现在的虚拟化技术对物理机（宿主机）的CPU资源消耗比例已经非常低了，

所以能够分配给虚拟机的CPU资源是比较充足的。一般来说，云平台都会有自己定义的CPU计算能力的单位（比如Amazon叫EC2 Compute Unit， ECU），然后给不同类型的云主机分配不同的计算资源。宿主机的CPU核数一般会比较大，因为CPU核数少就没有虚拟化的意义了。

从内存角度讲，宿主机也是高内存的，不同类型的云主机会分配到不同的内存，所以内存也不存在太大的问题。

现在很多系统的瓶颈都是在磁盘I/O上，虚拟化技术也不例外。哪怕宿主机用RAID提升磁盘I/O性能，性能提升还是有限。如果宿主机上的虚拟机过多，肯定会影响各个虚拟机的磁盘I/O性能。比如说，一个200核CPU的宿主机上有100个虚拟机在同时运行，那么平均分配到每个虚拟机的磁盘I/O性能只有宿主机的1%（当然所有的虚拟化技术都会进行磁盘读写的优化，所以实际上虚拟机的磁盘I/O性能要远远好于1%，但再好的优化也达不到所需要的性能）。因此，云平台都会有自己的调度算法来保证一个宿主机上的虚拟机个数不会超过可以容忍的上限。

总的来说，云主机的磁盘I/O肯定不如真正物理机上的磁盘I/O性能好，但在绝大多数场景下也是够用的。

性能的稳定性取决于以下两个因素。

- 云平台有没有做资源公平性的保证，有没有保证虚拟机对CPU、内存、磁盘的使用都有计量且是公平的。如果做了资源公平性保证，那么同宿主机内的虚拟机之间就不会相互影响使用。
- 云平台有没有超卖资源。所谓超卖是指宿主机实际只能支持100台虚拟机，但云平台卖了120台虚拟机。因为平均来说虚拟机的真正使用率并不高，所以云平台可以把用户平时多出来的资源再进行包装出去卖。但在最极端的情况下，用户的性能是得不到保证的。因此，一般的公有云平台都不会超卖，在私有云环境中超卖比较常见，因为可以明显减少企业内部的成本，而且企业内部各部门对极端情况有很好的容忍度。

总之，一般情况下，云主机的CPU和内存不是问题，其性能比一般的服务器要好。虽然磁盘I/O性能一般比不上物理机，但也基本够用。

搭建高可用服务

前面已经说了，云主机和普通服务器一样不可靠，所以用户不能只靠云主机来提供高可用的服务，原先在普通服务器上搭建高可用服务的方案都得移过来。

高可用服务最核心的思想是设计无状态的服务，不同团队有不同的实现方法。如果需要用户把原有的高可用方案都移植过来，工作量会相当大，而且有不少方案与网络直接相关，不能直接搬到云平台上。

一般的云平台都会提供负载均衡和高可用的数据库服务，用户利用负载均衡来实现Web服务的高可用，用云平台提供的数据库保证各种数据的持久化。以前用户在搭建高可用服务时，都会留下几台机器当备机，以防服务器出问题。而在云平台上，用户不需要购买备机，当有云主机出现问题时，申请一台顶上就可以。

需要注意的是，真正的云平台一定不是单机房的。单机房有可能出现各种意外情况，如整个机房断电。假如用户服务全部署在同一个机房的话，那么所有服务都将不可用，用户只能束手无策。所以推荐跨机房部署（Multi-zone）或者跨区域部署（Multi-region）。图1是Amazon zone-region概念图。

Multi-zone是指在相近的区域建设几个机房，每

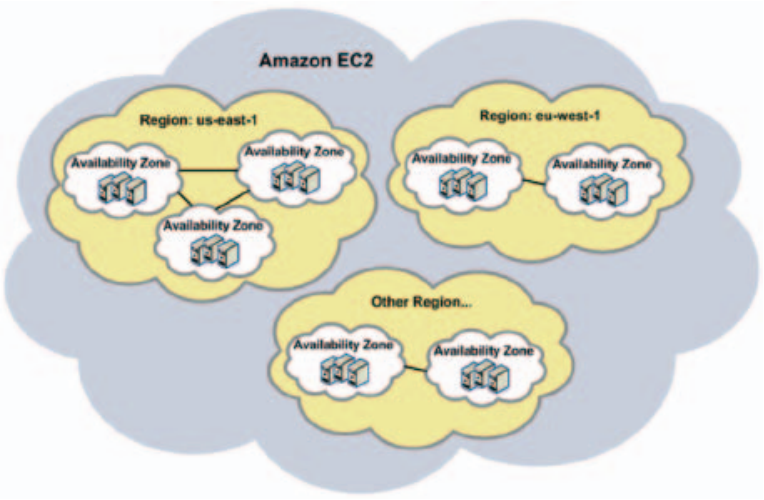


图1 Amazon zone-region概念图

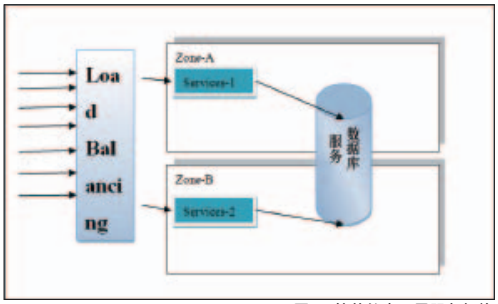


图2 简单的高可用服务架构

个机房用光纤连接，保证各机房之间的ping值小于1ms，同时保证各个机房之间有足够大的带宽，这样一个机房的数据可以快速迁移到另一个机房。Multi-zone基本解决了单机房不可靠的问题，很多云服务的可靠性都依赖它。

图2展示的是云平台上一个简单的高可用服务，其中云平台提供的负载平衡和数据库服务都是跨机房的。但由于政策的原因，目前国内没有一家公有云平台有Multi-zone的实现，但相信不久的将来会有公有云服务商提供类似的服务。

使用EC2/云主机应注意的问题

- 安全和监控是使用云主机时最需要注意的问题。云平台都有安全组（Security Group）的概念，给不同的服务组件启用不同的安全组，因此不要全都使用默认的安全组设置，只开启需要使用的端口即可。有些云平台可以使用密码登录管理，而有些只能用私钥登录，建议最好禁用密码登录，只允许使用私钥登录。云平台都有会有自带的监控体系，使用时最好启用监控。

- 制作镜像。申请一台云主机，安装好需要的软件，设置好服务配置环境，随后制作一个镜像，就可以用这个镜像批量生成同样的环境，减轻重复劳动以节省时间。同时，在某些云主机出问题时，可以快速搭建新服务器顶上，减少服务单点时间。

- 在不需要时结束云主机。需要注意的是，结束云主机与关机是两个概念。用户把云主机关机，云平台会继续对用户进行计费，因为用户可以随时重启机器，云平台必须保留用户的虚拟机，用户还在占用资源，因此还需要计费。在真不需要时，要把云主机结束，而不是简单的关机。结束云主机意味着会把这台云主机上的所有数据清空。如果用户觉得有必要保存这些数据，可以采用其他手段保存。

- 备份。永远不要把重要数据只保存到云主机的本地硬盘上。重要的数据可以放到EBS/云硬盘上，或者定期保存到云存储上。国内外这样的悲惨例子很多。在使用公有平台时，把重要的用户数据保存在云主机的本地磁盘，结果宿主机出问题了，数据全部丢失了。重申一次，EC2/云主机和物理机一样不可靠，重要的数据一定要备份。

EBS/云硬盘

前面讲到，云主机不可靠，会宕机、会丢硬盘数据、会不可提供服务。大多数用户能容忍死机，毕竟可以马上重新申请一台新机器，用镜像快速搭建服务器，几分钟后就可以提供服务了，或者做成高可用服务，根本不怕死机。但丢数据是用户不能容忍的，因为数据才是服务的根本所在。

虽然用户可以把数据保存到云平台提供的数据库和云存储上，但用户需求是多种多样的，云平台提供的服务经常不够用。能不能提供一种方案，满足各种产品对数据持久的渴望同时又满足各种产品对数据存储方式的需求呢？于是EBS应运而生了。其实EBS也不是什么新鲜东西。很早以前大家就开始使用iSCSI方案来解决文件的持久化存储问题，EBS不过是新瓶装旧酒。

如果要用通俗的话来解释EBS，那么移动硬盘的比喻是最容易让人理解的。云主机相当于用户购买了一台物理主机，EBS相当于用户购买了一块移动硬盘，把移动硬盘插到主机上，就可以使用。在这台主机出问题时，直接把移动硬盘拔了放到另外一台云主机上就可以继续使用了。EBS和移动硬盘的区别在于，移动硬盘可能会坏，而每份数据在EBS后台都保留两份以上的副本，保证数据不会丢失，所以EBS最大的作用是给云主机提供一块不会坏的磁盘。

前面讲了虚拟化的一个瓶颈在于磁盘I/O。而EBS的另一个作用是缓解宿主机的磁盘I/O读写的压力。因为EBS是基于网络的，如果云主机的用户把

所有重要数据都写到EBS上，其他数据写到本地磁盘上，那么相当于数据的写入多了一个很大的出口——网络，这大大缓解了宿主机本地磁盘读写的压力。而且宿主机的网卡不止一张，网络的吞吐量会超过本地磁盘的吞吐量，从而提高整个云主机的整体性能。

EBS不能独立于云主机而存在，正如移动硬盘离开了主机就没有使用价值了。同样，离开EBS单独存在的云主机也显然没什么优势了——不仅性能上会受到影响，如何低成本地将数据持久化保存也会成为一个难题。

EBS的性能到底如何呢？目前还没有标准答案。每家云平台的EBS实现并不一样，因此性能也不尽相同。但有两点比较明确：第一，目前EBS的性能不如本地磁盘；第二，EBS的性能取决于网络传输速度。

基于EBS，我们可以有更加廉价的高可用服务方案。如前面提到的高可用服务方案，都是需要使用负载均衡及至少两台云主机，费用会比较高。对服务停机时间有一定容忍性且对价格又比较敏感的客户来说，可以使用此方案。实施起来比较简单，将提供服务的软件环境做成镜像，将所有数据存储到EBS。当提供服务的云主机宕机时，根据镜像启动新的云主机，将EBS重新挂载上去，就可以直接恢复服务了。虽然宕机会耽误一些时间，但成本降低了。

EBS也有容量上限。目前，Amazon EBS的容量上限为1TB。之所以没有提供更大的容量，主要是因为空间太大对IO的性能影响比较大。

S3/云存储

如果说EC2和EBS解决了硬件层次的基础设施问题，让小公司也可以像大公司一样拥有丰富的IT资源，那么S3/云存储就是软件层次的基础设施，让小公司真正武装到牙齿，能够与资源丰富的大公司抗衡。

提到S3/云存储，不得不提Dropbox。Dropbox是小公司利用云计算发家致富，成为近100亿美元估值公司的经典案例。Dropbox使用S3/云存储作为后台存储，从而不用担心后台存储后院起火，不用做任何容量规划和容灾容错，因为这些问题Amazon都帮它解决了。这使得Dropbox可以专注于前端与客户端的开发。

为什么将S3/云存储当做基础设施的一部分呢？

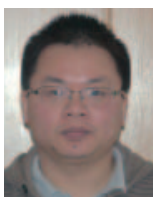
- S3/云存储解决了大规模数据待久化存储的问题。前面提到EBS虽然是持久化的，但有容量限制，最大容量为1TB。在这个信息爆炸的时代，每年产生的数据量会翻番。IDC报告宣称2012年全球新增的数据量会达到3ZB以上，如何保存海量数据成为一大难题。有S3/云存储后，用户可以把注意力集中到其他地方，更专注于业务而不用关心运维和容量规划。

- S3/云存储提供的是高持久化服务。Amazon S3的耐久性是99.999999999%，可用性是99.99%，因此可认为保存到S3上的数据不会丢。用户可以把数据库的备份保存到S3/云存储，同时也把数据库的操作日志随时保存到S3/云存储，那么在需要对数据库进行灾难恢复时，用户可以有99.999999999%的可能性保证数据可以恢复到特定的时刻。

S3/云存储最大的特点是无限容量、高持久化、高可用，但它是一个Key-Value结构的存储。与EBS相比，它缺少目录结构，所以在用户的业务里，一般都会使用数据库保存在S3/云存储上数据的元信息。

理想的S3/云存储应该跨机房部署，也就是说可以跨机房访问。但目前国内的云存储服务商都还没有实现。

综上所述，EC2/云主机和EBS/云硬盘组成了云计算平台硬件的基础架构，S3/云存储是云计算平台软件层次的基础架构，在它们的基础上，可以搭建出其他云计算产品。对于一个任何一个公有云平台，这三者缺一不可。



郭理靖

盛大云计算云数据库产品部经理，负责开发MongoIC、数据库云等云端的数据库服务。曾就职于金山、淘宝等公司。

责任编辑：杨爽（yangshuang@csdn.net）

C++ AMP异构并行编程解析

文 / 陈冠诚

C++ AMP（Accelerated Massive Parallelism）是微软于2011年6月推出的异构并行编程框架，在2012年2月的GoingNative大会上，微软正式发布了它的开放规范，并在开发工具中正式提供了对它的支持。

C++ AMP的目标是降低在由CPU和GPU共同组成的异构硬件平台上进行数据并行编程（data parallel programming）的门槛。通过C++ AMP，开发者将获得一个类似C++ STL的库，这个库将作为微软concurrency namespace的一部分，开发者既不需要学习新的C++语法，也不需要更换编译器就能够方便地进行异构并行编程。

C++ AMP设计原则

随着CPU由单核向多核转移，多核计算成为近几年的热点。另一方面，GPU编程也经历着一场变革。传统意义上，GPU一直是作为图形图像专用处理器而存在。然而，因为GPU拥有比CPU还要强大的浮点并行运算能力，我们是不是能让GPU来完成一些通用计算任务呢（例如科学计算中就需要大量用到浮点计算）？答案是肯定的。在这样的背景下，我们可以将并行计算从单纯地在多核CPU上做，扩展到在多核CPU与GPU共同组成的异构硬件平台上来。除了多核与GPU通用计算的快速发展之外，云计算更成为软件开发的一个重要趋势。实际上，云端的每一台服务器都可以是由多核CPU和GPU共同组成的异构硬件平台。微软的Herb Sutter介绍说：“我们认为多核编程、GPU编程和云计算根本不是三个独立的趋势。实际上，它们只是同一种趋势的不同方面，我们把这个趋势叫做异构并行编程”。进行异构并行编程需要一个统一的编程模型，这就是微软推出C++ AMP的原因。

微软决定另起炉灶，推出C++ AMP这样一个全新的异构并行编程模型的原因很简单，他们认为这个编程模型必须同时具备下面6个特征，而目前已有的CUDA和OpenCL不能同时满足。

- C++而不是C：这种编程模型应该利用好C++丰富的语言特性（例如抽象、模板、异常处理等），并且不会牺牲性能，因此不能像OpenCL一样只是C语言的一种方言。
- 主流：这个编程框架应该能被成千上万的开发者所使用，而不是只被少数人所接受。一个立见分晓的检验办法是：用该编程框架实现GPU上的Hello World是只需要几行代码，还是需要几十行才行？
- 最小的改动：这个编程模型应该只需要在C++上进行最小的改动就能够实现应有的功能。通过一个非常小、具有良好设计的语言扩展，我们就可以把绝大部分复杂的实现交由运行时系统/库去完成。
- 可移植的：这种编程模型应该让用户只需要一个二进制可执行文件就可以在任何厂商的GPU硬

件上运行。目前微软使用Direct Compute来实现在所有支持DX11的GPU上编程，但未来微软将会根据用户的需求在其他异构硬件平台上做相应的实现。

- 通用且不会过时：C++ AMP目前针对的是GPU并行计算。但微软希望，将来C++ AMP的程序可以无缝地扩展到其他形式的计算单元上去，例如FPGA、云端的CPU/GPU处理器等。
- 开放：微软将把C++ AMP做成一个开放标准，鼓励第三方在任何硬件和操作系统上实现C++ AMP编译器和运行时系统。目前AMD和NVIDIA都已声明将会支持C++ AMP。

C++ AMP介绍

下面通过一个简单的程序来了解一下C++ AMP的语法规则。首先我们需要引用amp.h这个头文件。C++ AMP中的模板都在concurrency这个命名空间内，所以也需要引用。在C++ AMP中主要有array和array view两种数据容器。这两者主要的区别在于array类型的数据在创建时会在GPU显存上拥有一个备份，在GPU对该数据进行完运算之后，开发者必须手动将数据拷贝回CPU。与之相比，array view其实是一个数据结构的封装，只有在它指向的数据被GPU调用时才会被拷贝到GPU上进行相应的计算。从下例中我们看到，声明array view数据时需要提供两个模板参数：array view元素的类型和数据结构的维度。因为aCPP、bCPP和sumCPP都是一维数组，因此我们在声明时传入int和1两个参数。

接下来就是最重要的计算部分了。parallel_for_each这个方法就是执行在GPU部分的代码入口。可以看到，parallel_for_each有两个参数，第一个名为sum.extent的参数是用于描述并行计算拓扑结构的对象。通过这个变量，我们指定有多少个GPU线程来并行执行该计算任务，以及这些线程的排列方式。sum.extend可以理解为按照sum的数据维度来分配相应数目的GPU线程。Parallel_for_each的第二个参数是一个名为“[=](index<1> idx) restrict(amp)”的lambda表达式。方括号里的“=”代表了lambda表达式的捕获列表。具体来说“[=]”表示lambda里捕捉的变量按照传值的方式来引用。该for循环的主要参数就是index<1> idx了，它其实代表的是GPU线程的编号。因为之前我们已经通过sum.extent定义好了GPU线程的数量和拓扑结构，所以这个index参数代表的就是一维的数组，即从0到4共5个数。最后一个参数restrict(amp)用来表示parallel_for_each的函数体运行在默认GPU设备上。当然我们也可以定义出amp之外的其他语法约束，具体内容请大家阅读参考文献[1]。在这之后就是循环体了。这个例子的循环体非常简单，就是让GPU用5个线程并行地把数组a和b中的元素依次相加并存到sum数组中去。

```
#include <amp.h>
#include <iostream>
using namespace concurrency;

void CampMethod() {
    int aCPP[] = {1, 2, 3, 4, 5};
    int bCPP[] = {6, 7, 8, 9, 10};
    int sumCPP[5] = {0, 0, 0, 0, 0};

    array_view<int, 1> a(5, aCPP);
    array_view<int, 1> b(5, bCPP);
    array_view<int, 1> sum(5, sumCPP);

    parallel_for_each(
// Define the compute domain
        sum.extent,
// Define the code to run on each thread
// on the accelerator.
        [=](index<1> idx) restrict(amp)
        {
            sum[idx] = a[idx] + b[idx];
        }
    );

    for (int i = 0; i < 5; i++) {
        std::cout << sum[i] << "\n";
    }
}
```

CUDA、OpenCL与C++ AMP

其实在C++ AMP之前已经有了两个异构编程框架：CUDA与OpenCL。CUDA（Compute Unified Device Architecture）是显卡厂商NVIDIA于2007年推出的业界第一款异构并行编程框架。在NVIDIA的大力支持下，CUDA拥有良好的开发环境、丰富的函数库、优秀的性能。但CUDA只能被用于在NVIDIA的显卡上进行异构编程，有先天的局限性。OpenCL（Open Computing Language）是业界第一个跨平台的异构编程框架，是Apple

领衔并联合NVIDIA、AMD、IBM、Intel等众多厂商于2008年共同推出的一个开放标准，由单独成立的非营利性组织Khronos Group管理。与C++ AMP类似，OpenCL作为一个开放的标准，并不局限于某个特定的GPU厂商，从这点上来看，NVIDIA自己独家的CUDA显得很封闭。我们可以把OpenCL在异构编程上的地位与OpenGL和OpenAL类比，这两个标准分别用于三维图形和计算机音频。

因为CUDA和OpenCL比C++AMP更接近硬件底层，所以前两者的性能更好，然而C++ AMP的易用性却优于CUDA和OpenCL。与C++ AMP基于C++语言特性直接进行扩展不同，OpenCL是基于C99编程语言进行的相关修改和扩展，因此C++ AMP比OpenCL拥有更高层次的抽象，编程更加简单。在CUDA和OpenCL中，kernels（运行在GPU上的代码）必须被封装成特定函数，而在C++ AMP中，代码看起来整洁得多。我们只需要使用for循环中内嵌的lambda函数就能完成异构并行计算，而且它的内存模型也在一定程度上被大大简化了。

那么在OpenCL、CUDA 与C++ AMP之间，开发者该如何选择呢?

如果你只需要在Windows平台上进行异构编程，并且看重易用性的话，C++ AMP无疑是最好的选择。依托于Visual Studio这个强有力的开发工具，再加上基于C++这一更高层抽象带来的先天优势，C++ AMP将为Windows开发者进行异构编程提供良好的支持。

如果你只需要在NVIDIA的GPU卡上进行异构编程，并且非常看重性能的话，CUDA应该是第一选择：在NVIDIA的强力支持下，CUDA在NVIDIA硬件上的性能一直保持领先，同时CUDA的开发环境也非常成熟，拥有众多扩展函数库支持。

如果你更注重不同平台间的可移植性，OpenCL可能是目前最好的选择。作为第一个异构计算的开放标准，OpenCL已经得到了包括Intel、AMD、NVIDIA、IBM、Oracle、ARM、Apple、RedHat等众多软硬件厂商的大力支持。

其实从编程语言的发展来看，易用性往往比性能更加重要。从Java和.NET的流行，到脚本语言的崛起，编程效率无疑是最重要的指标，更不用说开发者往往可以通过更换下一代GPU硬件来获得更好的性能。从这点来看，C++ AMP通过降低异构编程的难度，实际上也是推进了异构编程的普及。

总结

目前整个业界的异构硬件体系结构仍然处于快速演变之中。可以看到，许多厂商的处理器正在尝试融合CPU和GPU（例如AMD的Fusion、Intel的Larrabee和NVIDIA的Tegra3都融合了CPU和GPU）。如果将来的处理器上集成了CPU和GPU，并通过同一条总线使它们与内存直接相连，我们就不再需要像今天这样把数据在CPU和GPU之间搬来搬去了。随着异构硬件的发展，与之相对应的异构编程框架也需要随着演变。可以预见，今天我们看到的CUDA、OpenCL和C++ AMP都只处于一个初期形态，将来它们还会有很多新的变化。但有一点我们可以肯定：将来的异构编程一定会比现在更加容易。

参考文献

[1] Overview of C++ Accelerated Massive Parallelism
http://msdn.microsoft.com/en-us/library/hh265136.aspx

[2] C++ AMP实战：绘制曼德勃罗特集图像
http://www.cnblogs.com/Ninputer/archive/2012/01/03/2310945.html



陈冠诚
IBM中国研究院研究员，主要研究方向为并行计算、计算机体系结构及大规模分布式系统。个人博客为http://parallellabs.com。

责任编辑：卢鹃翔（ludx@csdn.net）



Windows 8的Metro应用（中）

从应用程序模型到文件系统，Windows 8在很多方面都做了实质性的改进，本文将重点介绍支撑Metro应用的WinRT，并以COM为线索介绍它的前世今生。

主持人：张银奎

《软件调试》一书作者，从事软件开发和研究10余年，对IA-32架构、操作系统内核、虚拟技术，尤其对软件调试有较深入的研究。翻译（合译）作品包括《数据挖掘原理》、《机器学习》、《人工智能：复杂问题求解的结构和策略》、《观止——微软创建NT和未来的夺命狂奔》等。

Windows 8 Beta版（消费者预览版）如期发布，这标志着Windows 8又顺利向前迈进了一步。虽然有人说，Windows 8可能重蹈Vista的覆辙，上市即死，部分理由是Metro应用丑而无用，新功能不足。但我觉得说这些话的人颇有些主观臆断。事实上，Windows 8在很多方面都做了实质性的改进，从应用程序模型到文件系统，从普通消费者在意的用户体验到专业人员看重的内核调试支持，只不过出于低调和谨慎，很多内部改变没有大肆宣传罢了。不轻信，不迷信，耳听为虚，眼见为实，我们还是继续用调试器这把利剑拨开浮云看真相吧。本期我们将重点介绍支撑Metro应用的WinRT，溯其源头，我们且从COM谈起。

COM归来

COM是什么？字面讲就是组件对象模型（Component Object Model），略深一点说，就是建立在接口和对象技术上的一种软件协作规范，可以在不同软件模块之间以组件的形式进行交互。如果时光倒退15年，当时最热门的技术可能就是COM了。但从2000年开始，随着.NET技术的全面推进，COM慢慢被冷落了，尽管这个技术一直存在，而且一直在微软内外都有使用。

十几年之后，或许是阴差阳错，或许是命中注定，COM技术在Windows 8中王者归来，被重新发扬光大了。如果你像我一样，书柜中还有关于COM的旧书，那么是时候请这些老朋友站出来发挥些价值了。但对于大多数新入道的程序员朋友来说，这可能有点天方夜谭。

不管怎样，我们还是先简要复习一下COM。为了避免生硬晦涩，下面就以我的几本书为脉络吧。

以时间为顺序，首先要说的是《对象嵌入与链接技术高级编程技术》，这是我6本书中最老的一本，也是唯一标题中不带COM字样的。而这也恰好反映了COM的渊源。简单说，COM是为了解决OLE（Object Linking and Embedding）问题而开发的技术。OLE的第一个版本发布于1990年，是基于古老的在Windows 2.0（1987年）中引入的DDE（Dynamic Data Exchange）技术。DDE技术有很多问题，贻害至今，我们在《拯救挂死的PowerPoint》一文中曾经给出一个经典的例子。在OLE 2中，人们开发了一套新的以Windows对象为核心的技术来替代DDE，这就是COM。这本书的作者是在1992年第一次接触OLE 2的，真正开始开发的时间可能还略早。OLE 2的发布时间，大约是1993年。很长一段时间中，OLE 2都是以OLE 2 SDK的形式发布的。在1995年5月的《Dr. Dobb's》杂志上登载了一篇名为“Inside the OLE 2 SDK”的文章，专门介绍这个SDK的内容并解释了OLE 2的重要概念。

翻了一下我的光盘资料库，在一个16位压缩包中，居然发现了这个SDK的2.03版，时间为1994年3月17日。感谢Windows的伟大兼容性，双击运行其中的setup.exe，这个18年前的安装

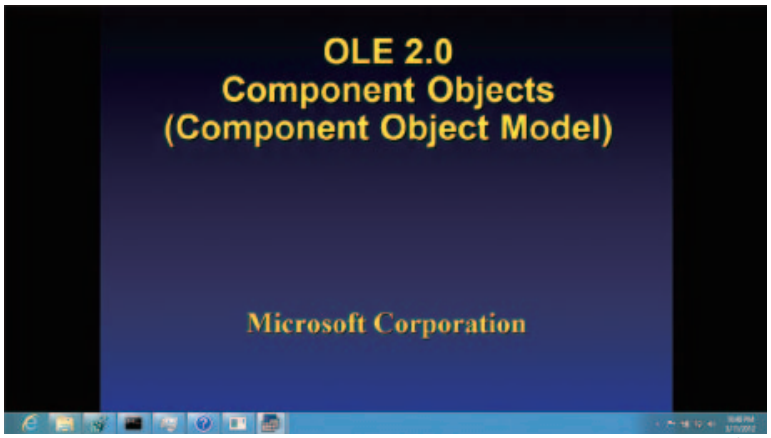


图1　作者电脑中的老PPT见证了COM技术在OLE 2中酝酿和成熟

包还工作得有板有眼。浏览安装好的内容，也井井有条，除了文档、示例、头文件、lib库、调试版本和发布版本的运行库外，还在一个名为slides的目录中附带了很多PPT文件和一个用于观看PPT的pptview.exe程序。它是真正的单文件绿色阅读器，只有981KB。双击运行，弹出一个选择文件的对话框，选择一个名为COM.ppt的文件，点击Show就打开了。

多好的用户体验，它代表了微软蓬勃发展时期的软件质量。巧得很，这个名为COM.ppt的演示文档作者正是我们在向大家介绍的这本“Inside OLE 2”的作者——Kraig Brockschmidt。文档的标题是“OLE 2.0 Components Objects (Component Object Model)”， OLE 2组件对象在上，COM在下，并放在括号中，这很好地说明了当时OLE 2与COM的关系。前者是名号，后者是实质。也预示着COM将要从后台的支持身份走向前台独立门户了。

第二本书是《COM技术内幕》，英文原名“Inside COM”，原书出版时间是1997年1月，中文版时间是1999年3月，我于同一年购入。此时的COM已经声名大噪，自成体系，是当时Windows操作系统（Windows NT和Windows 95/98）的一个基础部分。不过，无论是核心技术，还是主要函数的名称都还是源于OLE 2时代的，而且lib库和运行时DLL的名字直到今天还保留着OLE字样，分别叫ole32.lib和ole32.dll。

第三本书是《COM/DCOM技术内幕》，英文原名“COM/DCOM Unleashed”，出版时间是1999年3月，中文版的时间是2000年1月。这本书在亚马逊上的评价非常差，我也颇有同感，不过它刚好反映了后COM时代的特征——繁杂晦涩。为了向所谓的企业应用方向发展，很多概念大而空，缺乏实质。比如DNA（Distributed interNet Applications Architecture）、MTS（Microsoft Transaction Server）、MSMQ（Microsoft Message Queuing）、DCOM（Distributed COM）、COM+等。这些概念中最有价值的是DCOM，即所谓的分布式组件对象模型。DCOM是建立在RPC（Remote Procedure Call）基础之上的，目的是让COM组件对象可以跨越机器边界，因此有时又被称为ORPC（Object RPC）。DCOM从NT 4.0和Windows 98开始成为Windows的一部分，直至今日。

另外三本书包括潘爱民老师原创的《COM原理与应用》等，就不一一细说了。

WinRT出世

随着2000年.NET Beta版本的发布，特别是在2002年2月.NET 1.0正式发布后，COM热逐渐消退了。但正如前面的提到的，COM技术始终存在，而且一直在被使用着，只不过是没有什么发展罢了，因为微软把“砝码”都放在了.NET上。直到差不多10年后，COM技术在Windows 8中被“朝花夕拾”，重放光彩，这便是WinRT。就像理解和消化COM需要一段时间一样，理解WinRT也需要一段时间。

先解释一下名字，WinRT的全称是Windows Runtime，坦率地说，这个名字很不“给力”，词不达意，或许将来会被一个更好的名字所取代。至少，在这两个单词间加一个Object会好一些，也就是Windows对象运行时。Windows对象是OLE 2时代的一个常用语。在前面提到的《COM技术内幕》一书中，全书4个部分中的第1部分就叫Windows对象，近200页篇幅。其中特意对C++对象和Windows对象做了很多比较，写得非常精彩。作者使用旅行做比喻，说C++对象有语言和存取访问等限制只可以在国内（进程内）旅行，而Windows对象是很容易出国访问的，系统会帮

助这些对象处理各种签证和翻译事务。在一幅插图中，作者还预言某一天会开通更多的航线，让对象有更多的旅行目的地。诚然如此，10多年后，在新的WinRT框架下，COM对象的确有更大的活动空间了。

概括说，WinRT是建立在COM技术基础上吸取.NET先进特征的新型本地API系统。图2概括了它的关键特征。



图2　集大成者WinRT

我们将逐步解释和消化WinRT的这些特征。首先，WinRT代表了一种新形式的API，它改变了传统Win32 API的简单函数形式，让内部库以类和对象方式公开给应用程序使用，这一点与COM和.Net类似，但WinRT与传统COM的不同之处是创建对象的过程更简单了（借助编译器支持）。与.NET的差异是WinRT的对象都是以本地代码来实现的，这样的好处首先是高效，另外也可以非常自然地被C++这样的本地语言所调用。为什么反复强调“本地性”（Native）呢？推荐大家到Channel9网站上看一下前不久举行的Going Native 2012大会录像，听听包括C++缔造者Bjarne Stroustrup在内的多位大师是如何说的。

那么WinRT对象与COM对象有哪些异同呢，我们就以图3所示的WinRT对象为线索来简单比较一下吧，这幅图来自WinRT团队的开发经理Martyn Lovell在Build大会中题为“Lap around the Windows Runtime”的演讲。

首先，WinRT对象也都具有源于经典COM的IUnknown接口，一系列IStorage开头的接口看着也很眼熟，这都代表着与传统COM的密切亲缘关系。再看下面的两个绿色虚线方框，左侧框的

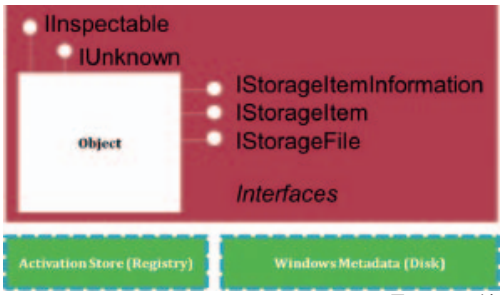


图3　WinRT对象

含义是用于激活WinRT对象的信息存储在注册表中，其实就是我们在2012年第3期提曾经到的ActivableClasses表键就属此类。右侧框的含义是描述WinRT API详情的元数据位于磁盘中，通常是以.winmd为后缀的文件名存储的。搜索一下，在Windows系统目录的System32\WinMetaData下能找到很多个winmd文件，是按API类别细分的，每个文件对应一个类，每个文件都不大，如图4所示。另外，在Windows 8的SDK中也包含了几个winmd文件，其中一个名为Windows.winmd文件，大小有1.31MB，应该是当前版本的所有WinRT API的合集。

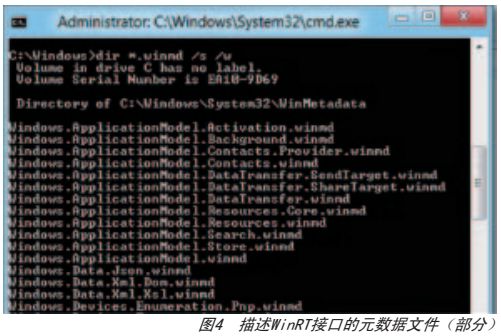


图4　描述WinRT接口的元数据文件（部分）

对于熟悉COM的读者来说，winmd文件与COM中的IDL（接口描述语言）文件的作用类似，但格式是二进制的，具有更容易支持不同开发语言和更好支持调试等优点。

从API的内容来说，WinRT涵盖了编写Metro应用程序所需的几乎所有方面，包括新外观的控件、用户交互、文件存储、网络、图形、视频、打印、安全等。

从编程语言的角度来说，WinRT支持C++、.NET以及JavaScript和CSS这样的网页脚本。不同语言编写的代码通过所谓的语言投射（Language Projection）技术对接到底层的WinRT运行时中，

本期问题：

上期答案：上一期的问题是DCOM和RPC的全程是什么，答案已经在本期的正文中出现，分别为Distributed COM（分布式COM）和Remote Procedure Call（远程过程调用）。

本期问题：图6中的RoEnableDebuggingForPackage函数的作用是什么？建议大家通过搜索找到解释和有关的工具后，最好再亲自动手试一下。

编者说明：

- 投稿信箱：
 contest@csdn.net
- 联系方式写在一个单独的TXT文件里，包括以下几项：
 1）姓名
 2）工作单位或学校
 3）电话联系方式
 4）邮寄地址
 5）E-mail地址
- 解答提交时间，最好早于当月15日。

也就是通常所说的二进制绑定。当然这需要编译器的支持，如果大家想试一下，那么建议下载刚刚发布的Visual Studio 11 Beta或者Visual Studio 11 Express的Beta版本。

图5归纳了WinRT的总体架构，也是源自刚才提到

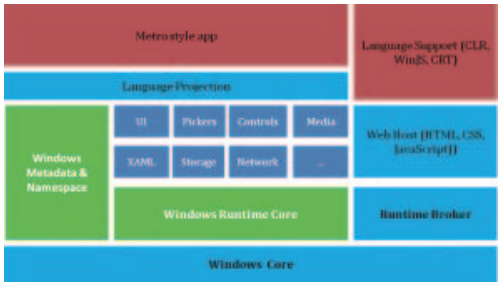


图5 WinRT架构

的Martyn Lovell的演讲。

从文件模块的角度来讲，图5中的Windows Runtime Core对应的是combase.dll，它在WinRT中的地位与ole32.dll在COM中的位置相当，而且有过之而无不及。随便启动一个Metro程序，比如Metro风格的控制面板，然后切换到传统桌面，使用WinDBG附加上去（可执行文件名SystemSettings.exe），执行lm vm combase便可以观察到这个文件的详细信息（略去部分信息）：

```
0:000> lm vm combase
    ImageSize:         0011E000
    File version:      6.2.8250.0
FileDescription:   Microsoft COM for
Windows
```

与ole32.dll导出了一系列Co开头的函数用以支撑COM技术类似，combase.dll也导出了一系列函数，名字以Ro开头，它们是支撑WinRT技术的关键所在，在WinDBG执行x combase!Ro[a-u]*便可

```
combase!RoGetParameterizedTypeInstanceIID = <no ty
combase!RoGetServerActivatableClasses = <no type i
combase!RoTransformErrorW = <no type information>
combase!RoOriginateError = <no type information>
combase!RoEnableDebuggingForPackage = <no type inf
combase!RoResolveRestrictedErrorInfoReference = <n
combase!RoInitialize = <no type information>
combase!RoActivateInstance = <no type information>
combase!RoGetExtensionRegistration = <no type info
combase!RoGetExtensionRegistrationByExtensionId =
combase!RoGetDesignMode = <no type information>
combase!RoParameterizedTypeExtraGetTypeSignature =
combase!RoSetErrorReportingFlags = <no type inform
combase!RoReportCapabilityCheckFailure = <no type
combase!RoGetActivatableClassRegistration = <no ty
combase!RoUninitialize = <no type information>
combase!RoRegisterActivationFactories = <no type i
combase!RoDisableDebuggingForPackage = <no type in
combase!RoFreeParameterizedTypeExtra = <no type in
combase!RoGetActivationFactory = <no type informat
combase!RoGetErrorReportingFlags = <no type inform
combase!RoSetDefaultBrowser = <no type information
combase!RoRevokeActivationFactories = <no type inf
combase!RoTransformError = <no type information>
combase!RoOriginateErrorW = <no type information>
```

图6 支撑WinRT的核心函数

以看到这些函数（图6）。

熟悉COM的朋友很容易可以把图6中的某些函数与以前的COM函数对应起来，比如RoInitialize对应于CoInitializeEx，RoUninitialize对应于CoUninitialize。但也有些函数的命名有所变化，比如在COM中，CoCreateInstance或者CoCreateInstanceEx函数用于创建对象实例，而在WinRT中，RoActivateInstance用于获取对象实例，也就是把名字中原来的Create改为了Activate。根据文档和我的观察，这个改动只是字面上的，实际执行的还是Create的操作。类似的字面变化还有，在COM中用于获取类工厂接口的是函数CoGetClassObject，而在WinRT中，用于获取类工厂接口（IActivationFactory）函数被命名为RoGetActivationFactory，但二者所承担的任务是一样的。图7给出了Metro应用在进程启动早

```
0:000> kn
 # ChildEBP RetAddr
00 006bf5b4 596010e2 combase!RoActivateInstance
01 006bf5cc 59601014 Windows_UI_Xaml!Windows::Foundati
02 006bf61c 5919a323 Windows_UI_Xaml!RunInActivationMo
03 006bf62c 003c6f92 Windows_UI_Xaml!DirectUI::Applica
04 006bf714 003c6a02 HiMetro!Windows::UI::Xaml::IAppli
05 006bf83c 0040d7cd HiMetro!Windows::UI::Xaml::Applic
06 006bf954 0048a3ef HiMetro!main+0x9d
07 006bf9a8 0048a088 HiMetro!_main+0x12f
08 006bf9b0 00487e5d HiMetro!WinMain+0x8
09 006bfa14 00487fbd HiMetro!__tmainCRTStartup+0x22d
0a 006bfa1c 75912b9d HiMetro!WinMainCRTStartup+0xd
0b 006bfa28 77cb7221 KERNEL32!BaseThreadInitThunk+0xe
0c 006bfa60 77cb71cd ntdll!__RtlUserThreadStart+0x4a
0d 006bfa78 00000000 ntdll!_RtlUserThreadStart+0x1c
```

图7 创建WinRT对象

期，主线程调用RoActivateInstance时的栈回溯。

图7中的函数调用关系与图5中所示架构层次刚好是对应的（上下颠倒）。HiMetro对应了图5中的应用层，Windows.UI.Xaml对应于WinRT的API组件层，combase对应于WinRT核心模块层。#5-#3栈帧表示HiMetro应用的main方法中调用Xaml API中Application类的静态方法Start，进入到了Windows.UI.Xaml模块中，两者之间因为都是本地代码，所以直来直去，非常高效。而后，Start方法继续调用模块内部的RunInActivationMode函数，开始Metro之旅，下一期将结合编译器（VS 11）继续介绍。

Windows 8应用开发概览

文 / 李建忠

在本文中，作者将从开发平台的角度近距离观察Windows 8，探寻这款新的操作系统将给软件开发行业带来怎样的变化，帮助读者了解Windows 8开发技术的方方面面。

Windows 8消费者预览版终于在2月29日发布。从官方角度来看，微软一直将Windows 8的重要性与Windows 95的历史地位类比，即所谓“Reimagines Windows（重塑 Windows）”。熟悉软件历史的朋友都知道，当年微软推出Windows 95时，许多在DOS平台上的软件公司（包括许多中国的公司）纷纷折戟沙场，那么微软在Windows 8中有哪些革新？这些革新将给业界带来什么样的生态变化？

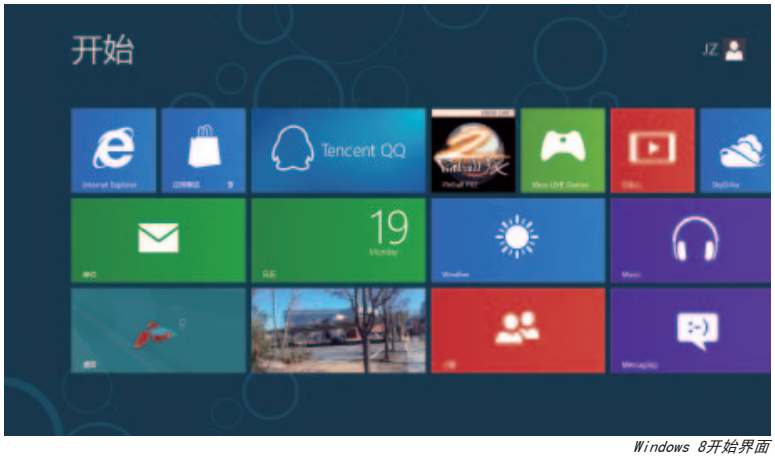
带着这些问题并结合自己一段时间以来把玩研究Windows 8的体会，我将带领大家从开发平台的角度近距离观察Windows 8。我将Windows 8的革新划分为4个关键词：“体验”、“开发”、“运行”、“部署”。下面我将围绕这4个关键词展开阐述，帮助大家了解Windows 8开发技术的方方面面。

体验：触控优先的Metro用户体验

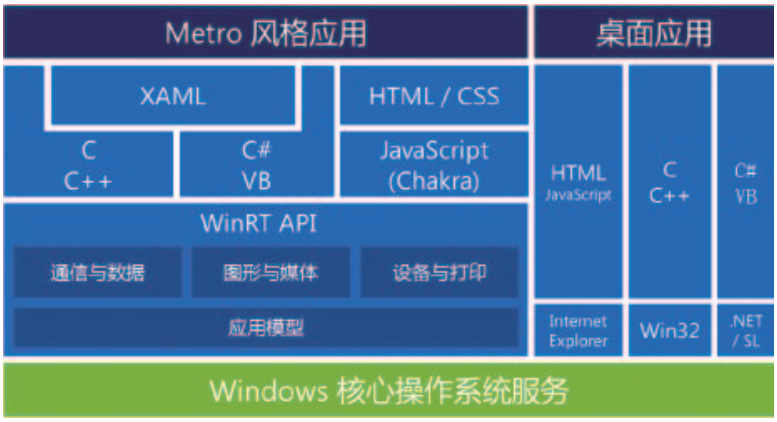
毫无疑问，“多点触摸”已经成为新一代计算设备的标配，它是继键盘、鼠标之后意义最深远的人机交互方式。虽然微软早在2008年就开发出桌面多点触摸设备Surface，但没有在Windows 中很好地融入触摸体验一直让微软饱受业界诟病。虽然Windows 7也支持触控，但基本上只是将触控模拟为鼠标操作，体验很糟糕。而Windows 8的设计从一开始就将“触控”作为第一人机交互方式进行考虑。

但另一方面，Windows 8又非“唯触控论”者，Windows 8希望能够统一用户在PC、平板、甚至手机这3种设备之间的体验，整合Metro应用与传统桌面应用，兼顾专业用户（对鼠标和键盘的精准操控需求）和消费用户（对直接触摸的操控需求）等多种场景下的需求。

在“触控优先、兼顾键盘和鼠标”的目标指导下，微软制定了一系列提升Windows 8用户体验



Windows 8开始界面



Windows 8系统结构

的“Metro设计原则”：例如“凸显内容，去除边框等修饰”，“使用活动磁贴支持应用动态通知”，“使用语义缩放支持多种内容视图”等。关于这方面的详细规范，大家可以参考MSDN文章《规划Metro风格应用》。

这些Metro设计原则在开发具体应用时，必须得到遵守，否则有可能无法通过Windows应用商店的审核。微软也在Visual Studio 11 Beta版中，对Metro设计原则给予了充分支持，使得开发人员能够方便地按照Metro规范行事。

开发：兼顾性能与复用的开发框架

在应用程序开发方面，为了支持Metro应用的体验、运行、部署等需求，Windows 8在原有桌面应用开发框架之外，推出了以WinRT（Windows Runtime）为基础的开发框架。WinRT的设计目标是：兼顾性能（操作系统基础组件必须考虑性能）和复用（多语言编程之间的组件复用），维护微软经营多年的3大技术资产：以C/C++为主的COM资产，以C#、VB.NET为主的.NET资产，和以JavaScript、CSS/HTML为主的Web资产。对WinRT我有以下3点解读。

■ WinRT的核心服务是一个以COM为基础的轻量级互操作方案，以此支持C++、.NET与JavaScript几种不同语言之间的混合编程。这在WinRT中被称为语言映射（Language Projection），这种映射有两个特点：一方面将互操作规则内置在编译器中，因此无需程序员像传统.NET与Native代码互操作那样做烦琐的Attribute声明（有过P/Invoke或COM互操作痛苦经验的朋友应该对此深有体会），只需简单地了解一些基本规则即可。另一方面：由于采用基于COM组件物理地址的绑定，因此这种互操作调用性能极高。

■ 通过WinRT，微软提供了一系列支持Metro应用的组件类库，包括：应用模型、通信与数据、图形与媒体、设备与打印等。基于基础设施的性能要求，这些组件主要以C/C++本地代码编写，然后通过WinRT的语言映射机制，可被C++、.NET与JavaScript几种不同语言复用。需要指出的是，基于Metro应用移动场景的灵敏响应需求，WinRT中很多组件提供了异步编程接口，这在不同语言中有不同的设计模式或语言机制来支持，这是开发者在WinRT下编程的一项关键技能。

■ 在界面描述方面，WinRT支持两种界面描述语言：XAML和HTML/CSS。XAML是微软在WPF和Silverlight领域经营多年的资产。之前很多朋友一直纠结微软是否在Metro中支持WPF或Silverlight？这个问题在微软官方回答中也是模棱两可。但我认为这个问题要从两个角度来看。如果纯粹讲WPF或Silverlight的应用模型或组件是否在Metro中依然得到支持？答案当然是否定的。但如果从XAML的角度看，WPF/Silverlight中XAML的各项机制在Metro中又是延用的，并且同时支持.NET语言和C/C++操作。由于HMLT5的界面描述能力大幅提升，HTML也被WinRT作为第一等界面描述方案（通过JavaScript来操作），这也是Windows 8拥抱Web的具体体现，使用现有的Web开发技能（HTML5/CSS/JavaScript），开发人员完全可以开发出第一等Metro应用。从历史角度来看，微软一向对自己不能控制的语言抱有戒心，但这次微软在Windows 8中如此积极拥抱Web开放技术，反映了微软心态的变化。我认为，这是微软极具气度和智慧的一次选择。

我认为，WinRT总体是一个“新瓶装旧酒”的方案。微软在WinRT的开发中抛弃了以前的“技术美学”，而选择了更加实用主义的道路——如果从技术美学角度来看，WinRT有点七拼八凑的感

觉，但这种兼顾了性能与复用的方案对于Metro应用开发非常有帮助。WinRT本身没有创造太多新东西，但通过巧妙的“粘合方案”（即语言映射），很好地利用了微软经营多年的几大技术资产，以此来支持不同背景的程序员快速开发Metro应用。

运行：面向移动场景的应用执行环境

由于Winndows 8主要面向移动场景，操作系统要考虑高响应和低功耗等需求，在应用执行环境方面，Metro应用与传统Windows或.NET应用也有很大区别。所有这些变化，让开发人员在编写Metro应用时面临的新挑战。在这方面，我有以下几点解读。

典型的Metro应用程序有三个状态：运行、挂起、终止。应用程序挂起时，所有线程都挂起，而Windows内核会确保应用不会在可能导致大量死锁的关键区挂起。此时应用仍然保留在内存中（从而支持应用的快速唤醒），但不占用CPU、磁盘或网络（降低功耗）。当用户将应用调回前台时，立即从挂起状态恢复到运行态。应用程序终止除了用户触发的关闭操作外，执行环境也会根据当前环境资源需求情况（例如系统需要更多内存时），自主决定。

对于一些需要运行在后台的特殊应用需求，例如：后台音频播放，后台上传下载，实时通信应用等，Metro执行环境给予了特殊处理，开发者需要针对这些特定应用进行声明，从而获得特殊的后台执行能力。

对于一些需要即时动态通知的应用，Metro支持桌面图标的活动磁贴功能，从而满足在低功耗的前提下，保持移动在线应用的描述能力。这也是应用程序状态的一种创新管理方式。

由于信任边界的管理，Metro需要对应用进行信任隔离。但同时，Metro又通过Contract来实现一些特殊的应用需求，例如应用之间共享数据，将不同应用连接起来，提供上下文相关的设置方式等。用好Contract是实现Windows流畅体验的一个关键。

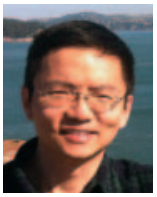
部署：便捷访问Windows应用商店

应用商店（App Store）模式已经成为移动应用部署的标准模式，Windows 8在这方面的支持（即Windows Store）也当仁不让。一方面，微软为Windows Store的应用发布、管理、统计提供了非常简便高效的方案。另一方面，微软设定了非常灵活的商业模型如免费、收费、应用内支付、试用版、第三方应用和广告支持等。在分成模式方面，微软也做了巨大让利，对于新应用，开发者分成比例为70%；如果应用收入超过2万5千美元，分成比例达则到80%。



Windows 8应用商店

我们必须承认，微软在应用商店方面是个后来者，但相信凭借Windows操作系统强大的投射能力，微软经营多年的开发商生态环境，强大的快速开发工具，以及巨大的让利空间，Windows 应用商店将成为移动开发商必须重视的一个应用分发渠道。



李建忠

PM-China、祝成科技创始人。2010年获微软区域技术总监荣誉称号，译有《Microsoft .NET框架程序设计（修订版）》。

微博：@PM-China李建忠

责任编辑：卢鹃翔（ludx@csdn.net）

Twitter Bootstrap：前端框架利器

文 / 廖伟华

Web前端开发者每天都与HTML、CSS、JavaScript打交道，然而不少人都是周而复始地写模板、样式和交互效果，并没有想过如何将这些重复的工作整合在一起。Twitter推出的Bootstrap能够帮助Web前端开发者摆脱这种重复劳动。

Bootstrap的历史

为了应对复杂的需求，早期的Twitter前端工程师在开发网站时几乎采用了所有自己熟悉的前端库。造成了网站维护困难、扩展性不强、开发成本高等问题。此时BootStrap被提上了日程。Twitter要求前端工程师完全依靠这一单一框架进行前端开发。

Twitter在2011年8月将其开源，并在2012年2月3日发布了2.0版。在GitHub上，这个项目已有拥超过2万位关注者和4000个分支。Bootstrap的设计者、著名前端工程师Mark Otto这样写道：“Bootstrap是我和Jacob Thornton编写的一个前端工具箱，目的是为了帮助设计师和Web前端开发人员快速有效地创建一个结构简单、性能优良、页面精致的Web应用。它使用了最新的浏览器技术，可以提供精致的网页排版方式以及表单、按钮、表格、网格栅格化、导航等诸多元素。”Bootstrap的内置样式继承了Mark Otto简洁亮丽的设计风格，任何开发团队都能使用它提供的HTML模板、CSS样式和jQuery组件来布署或者重建一个外观漂亮的页面应用。

BootStrap 2的新特性

BootStrap 2在原有特性的基础上着重改进了用户的体验和交互性，比如新增加的媒体展示功能，适用于智能手机上多钟屏幕规格的响应式布局，另外新增了12款jQuery插件，可以满足Web页面常用的用户体验和交互功能。

BootStrap 2的运用

Bootstrap的文件结构

读者可以直接从GitHub下载到Bootstrap源码，本地解压后可以看到这样的目录结构：docs、img、jquery-ui-bootstrap、js和less。其中最为重要的是“docs”下的CSS样式文件，“less”中的编译文件和“js”下的jQuery插件。

Bootstrap的安装

关于如何应用提取出来的文件大致有两步：第一是如何安装Bootstrap的基本样式，第二是如何调用Bootstrap的jQuery插件，我们首先来看样式的安装。

样式的安装有多种方法，图1展示的是一种常用的调用样式方法“link”。

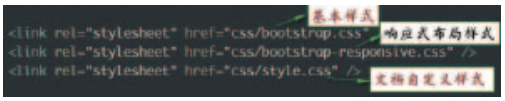


图1 Bootstrap样式调用方式

这里有两个关键点，其中“bootstrap.css”是Bootstrap中的基本样式文件，只要使用Bootstrap就必须调用这个文件。而“bootstrap-responsive.css”则可以根据你的爱好来选择，如果想让项目具有响应式布局的效果，就必须要调用这个样式文件，而且调用必须遵循先后顺序，“bootstrap-responsive.css”必须放置在“bootstrap.css”之后，否则便不具有响应式布局功能。而最后的“style.css”是项目中的自定义样式，用来覆盖Bootstrap中的一些默认设置，便于开发者定制。

“CSS”样式安装完后，就可以进入“js”的调

用，方法很简单，只需把想要的jQuery插件按照与上一步相似的方式加入到代码中。

Bootstrap 2的模块

BootStrap 2的模块从大的方面可以分为布局框架、页面排版、基本组件、jQuery插件以及变量编译的Less几个部分。与第1版相比，Bootstrap 2增加了多个新模块，比如布局框架中的“响应式布局”，页面排版中的“ICON”，基本组件中的“进度条”，而jQuery插件从以前的5个效果增加到12个，完全可以满足项目常用的交互效果。下面来简单了解一下Bootstrap 2中各模块的功能。

页面布局

布局在每个项目中都必不可少，Bootstrap在960gs的基础上扩展了一套优秀的Grids布局，而在“响应式布局”中有更强大的功能，能让网格布局适应各种设备。使用也相当简单，只需要按照HTML模板应用，就能轻松地构建你所需的布局效果。此外，改变模板中的类名，就能实现不同的布局风格。比如常见的“固定布局”，只需要在HTML中添加“container”类名；而要实现“流体布局”，只需要在HTML中添加“container-fluid”类名。Bootstrap还为开发者设计了“Responsive”， 令布局框架更为出色。开发者可以在此基础上进行任何样式的覆盖，从而实现理想中的响应式设计。

页面排版

页面排版的好坏直接影响产品风格，说直白点就是好不好看。在Bootstrap中，页面的排版都是从全局的概念上出发，定制了主体文本、段落文本、强调文本、标题、Code风格、按钮、表单、表格等格式。而Bootstrap 2中又添加了几个新亮点。其一是“Code”使用了“Google Prettify”插件，增强了代码的阅读体验；其二在“按钮”中增加了组合、下拉、图标等效果，如图2所示。

第三就是“Icon”的使用，Bootstrap在“Icon”部分采用了“Sprites”技术，为大家准备了上百种常用的“Icon”图标应用。

说起Icon，这里有必要在向读者推荐一个从BootStrap扩展出来的Font Awesome项目。它是

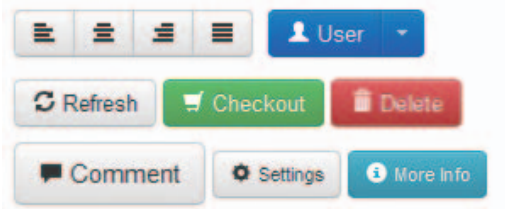


图2 Bootsrap 2中的按钮样式

Dave Gandy在Bootstrap的基础上扩展出来的一个Icon主题，最大的特点在于，整套图标中没有运用任何图片。大家可能会觉得奇怪，没有图片如何制作Icon？难道是通过纯CSS编写的吗？是的，Font Awesome项目中主要运用了CSS3的“@font-face”和“伪元素”一起实现。

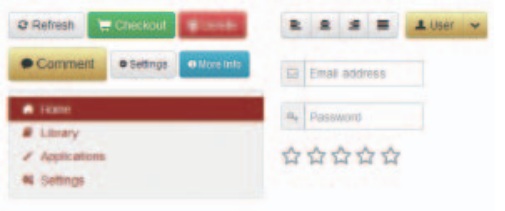
要使用Font Awesome，首先需要在服务器上安装“fontawesome-webfont”字体，接着在样式中通过“@font-face”来启用这些字体：

```
@font-face {
    font-family: 'FontAwesome';
    src: url('../font/fontawesome-webfont.eot');
    src:
    url('../font/fontawesome-webfont.eot?#iefix') format('embedded-opentype'),
    url('../font/fontawesome-webfont.woff') format('woff'),
    url('../font/fontawesome-webfont.ttf') format('truetype'),
    url('../font/fontawesome-webfont.svg#FontAwesomeRegular') format('svg'),
    url('../font/fontawesome-webfont.svg#FontAwesomeRegular') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

具备了上面的条件后，就只要样式中调用“FontAwesome”字体，并在“伪类元素”中使用对应的“字符编码”：

```
.icon-glass:before          { content: "\f000"; }
.icon-music:before          { content: "\f001"; }
.icon-search:before         { content: "\f002"; }
.icon-envelope:before       { content: "\f003"; }
.icon-heart:before          { content: "\f004"; }
.icon-star:before           { content: "\f005"; }
.icon-star-empty:before     { content: "\f006"; }
```

这样一来，就不采用任何图片实现Icon效果，一起来看一个效果图吧。



基本组件

基本组件是Bootstrap 2的精华之一，里面都是开发者平时需要的交互组件。比如“网站导航”、“Tabs”、“工具条”、“面包屑”、“分页栏”、“提示标签”、“产品展示”、“提示信息块”和

“进度条”等。这些组件都配有“jQuery”插件，运用它们可以大幅度提高用户的交互体验，使产品不再那么呆板无吸引力。说了这么多种组件，具体该如何使用？下面我以常见的“Tabs”组件为例，向大家展示使用方法。

要想使用BootStrap基本组件，必须满足三点：第一，最基本的HTML结构要对号；第二，需要Bootstrap中的“jQuery”插件提供相应功能；第三，在项目中对应的“Tabs”元素上启用“Tabs”功能。

1. HTML Markup

```
<ul class="nav nav-tabs">
  <li class="active"><a href="#home">Home</a></li>
  <li><a href="#profile">Profile</a></li>
  <li><a href="#messages">Messages</a></li>
  <li><a href="#settings">Settings</a></li>
</ul>

<div class="tab-content">
  <div class="tab-pane active" id="home">...</div>
  <div class="tab-pane" id="profile">...</div>
  <div class="tab-pane" id="messages">...</div>
  <div class="tab-pane" id="settings">...</div>
</div>
```

2. 调用jQuery插件

```
<script src="js/bootstrap-tab.js"></script>
```

3. 开启“Tab”功能

```
<script>
  $(function () {
    $('.tabs a:last').tab('show')
  })
</script>
```

对于其他组件，使用方法相近，在此不做赘述。

jQuery插件

Bootstrap中的jQuery插件主要用来帮助开发者实现与用户交互的功能，在第1版中，Bootstrap就为大家提供了6种常见的插件。

弹出框（Modals）：在JavaScript模板基础上自定义的一款流线型、灵活性极强的弹出蒙板效果的插件；

下拉框（Dropdowns）：Bootstrap中一款轻巧实用的插件，可能帮助你制作具有下拉功能，比如下拉菜单、下拉按钮、下拉工具条等效果；

滚动条（Scrollspy）：实现滚动条位置的效果，比如在导航中有多个标签，用户点击其中一个标签，滚动条会自己动定位到导航中标签对应的文本位置；

Tabs：这个插件能够快速实现本地内容的转换，动态切换标签对应的本地内容；

提示工具（Tooltips）：是一款优秀的jQuery插件，无需加载任何图片，采用CSS3新技术，动态显示“data-attributes”存储的标题信息；

提示面板（Popover）：在Tooltips的插件上扩展，用来显示一些叠加内容的提示效果，此插件需要配合Tooltips一起使用。

Bootstrap 2在前面6种插件的基础上又新增加了6种jQuery插件：

警报信息（Alert）：用来关闭警报信息块；

按钮（Button）：用来控制按钮的状态或更多组件功能，比如“复选框”、“单选按钮”以及“载入状态条”等；

手风琴（Collapse）：一款轻巧实用的手风琴插件，可以用来制作折叠面板或菜单等效果；

幻灯片（Carouse）：实现图片播放功能的插件；

补全文本（Typeahead）：可以记住文本框输入的文本，下次输入时可以自动补全；

动画效果（Transitions）：BootStrap使用这个插件，为一些动效果增加了过渡性，使动画效果更细腻、生动。

上面简单介绍了Bootstrap 2中的jQuery插件，至于如何使用，还需要根据Bootstrap所提供的文档，以及各插件的参数，具体问题具体分析。因为只有充分了解，才能灵活运用。

变量编译Less

Less是一个动态CSS语言框架，基于自己的JS引擎或者服务器端对传统的CSS进行了动态的扩展，使得Less具有更强大的功能和灵活性。基于Less，我们编辑CSS就可以像使用编程语言一样，定义变量、嵌入声明、混合模式、运算等。Bootstrap中有一套编辑好的Less框架，开发者可以将其应用到自己的项目，也可以通过less.js、Less.app或Node.js等方法来转译Less文件。Less文件一旦编译，Bootstrap框架就仅包含CSS样式，这意味着没有多余的图片、Flash之类元素。Less好处是利于编写，但缺点是只适合用于开发一些简洁的CSS样式。举个简单的例子，如果你

想为多个样式设置同一个颜色，可以这样写。

```
@color: #FFFFFF;
h1 {
color: @color;
}
#sub{
color: @color;
}
```

编译后的CSS如下所示：

```
h1 {
    color: #ffffff;
}
#sub{
    color: #ffffff;
}
```

以上代码只是最简的Less应用示范。除了变量，还可以在CSS中进行数学运算、传参、文件相互引用、规则嵌套等。可以想象，这种编写CSS的方式会为前端开发带来多大的改变。无论是从灵活性、扩展性还是可维护性上，Less都让CSS开发效率有了大幅提升。

Bootstrap的jQuery UI

Bootstrap的jQuery UI其实是从框架中衍生出来的一个jQuery UI主题，受到Twitter项目的启发，Addy Osmani也在Bootstrap的基础上整理出一个jQuery UI Bootstrap主题。

jQuery UI Bootstrap除了包含Bootstrap各个方面功能之外，还在其基础上补充了以下特性：动态添加Tabs、日期范围选择组件、自定义文件载入框、滑动块、日期控件。

BootStrap 2案例

通过前面的介绍，你可能已经体会到Bootstrap的强大，但能否运用到生产项目中，可能还存有疑问。

目前使用Bootstrap的著名案例有NASA和MSNBC的Breaking News。此外很多CMS也在运用Bootstrap框架，比如大家熟悉的WordPress、Drupal等。如果你还想了解更多Bootstrap案例，可以参考Wrapbootstrap.com。

BootStrap 2的优势和缺点

如今的Bootstrap已包括了几十个组件，每个组件都自然地结合了设计与开发，具有完整的实例文档，定义了真正的组件和模板。无论处在何种技术水平的开发者，也无论处在哪个工作流程中，都可以使用Bootstrap快速、方便地构建开发者喜欢的应用。

难能可贵的是，Bootstrap依旧本着“并行开发”、“作为产品的风格指南”和“迎合所有的技能水平”的原则帮助开发者解决实际问题，不断完善自己，吸引更多人选择Bootstrap应用于自己的项目中。

然而古人云“万物相生相克”，有好就有坏，Bootstrap也是一样。对于在国内的开发者来说，最可怕的就是IE兼容问题。目前Bootstrap对IE6到IE8的支持都不友好。另一个缺点是，采用Bootstrap的模板，网站结构时常会显得臃肿。此外，覆盖一些样式时会造成代码冗余。但与其他前端框架相比，我个人觉得Bootstrap的缺点仅此而以，至于其他方面希望有机会与大家一起探讨和学习。

总结

Bootstrap是一套前端开发利器。它可以帮助我们加速项目开发，让我们身处在一个完备的系统中，拥有一致的设计和实现方法。不需要在外观上花费过多时间，使开发者能将精力集中于更重要的功能。

Bootstrap将改变我们的合作方式与开发进程，任何人都可以基于Bootstrap建立可扩展的前端工具包，或者在它的基础上启动属于自己的框架。



廖伟华

网名大漠，目前供职于上海慕和网络科技视觉设计部，从事Web前端和手机游戏工作，对Web前端、移动端、前端框架抱有浓厚兴趣。个人博客：www.w3cplus.com。

责任编辑：卢鹤翔（ludx@csdn.net）

使用BackboneJS搭建SPA

文 / 吕婷

BackboneJS自从2010年发布以来，受到了业界的广泛关注。“豆瓣说”和“豆瓣阅读（阅读器）”是两款以它为基础框架搭建的SPA，本文将结合这两款产品，向读者介绍BackboneJS的方方面面。

BackboneJS是由Document Cloud公司推出的开源项目，作者是Jeremy Ashkenas，初衷是为Web应用提供一个合理、简洁的基础框架。自2010年10月发布以来一直受到业界关注，拥有相当多的成功案例，包括我们熟悉的37signals、LinkedIn Mobile、Flow、Trello等，从内容发布到项目管理，从个人工具到地图应用，使用场景非常广泛。

在介绍BackboneJS之前我们先了解3个概念：1. Single Page Application；2. JavaScript MVC；3. JavaScript MV* Framework。

Single Page Application

SPA是一种Web应用，全部在浏览器端执行，它需要具备以下几个最基本的特点：

1. 使用过程中不需要向服务器请求新的页面；
2. 数据以JSON格式，通过异步请求（或本地存储）的方式进行读写；
3. 各个“页面”通过URL标识，以便资源分享；
4. 完整的浏览历史管理等。

需要使用的技术除了HTML/CSS/JavaScript/AJAX，还有JSON、前端模板引擎，以及组织整个应用的框架等。最经典的实例就是2004年新版Gmail，以及2009年的新版Twitter。

JavaScript MVC

对于后端开发者来说，MVC是一个熟悉的术语。在浏览器端应用开发中，我们借用了这一经典设计模式来设计基础框架，将应用中的View和Model分离，再通过Controller来协调双方，从而产生出更多元化、重用性更好的代码。

类似的，有些框架借用了MVC的衍生模式：MVP或MVVM，于是就产生了JavaScript MVC这一概念。

JavaScript MV*框架

做为前端开发者，在开发SPA时通常会选择一个JavaScript MVC框架。那么我们对于这样一种框架有哪些期望呢？主要有以下几种：1. Model与View的分离；2. Model与View之间的协调者或沟通机制（Model变动自动更新View）；3. 清晰合理的代码组织结构；4. 可测试的代码；5. 尽可能少的“耦合代码”。

除此之外，在功能上还需要具备路由功能、历史管理、与数据存储层的交互等。

基于以上这几点期望以及不同的设计模式、产品需求，甚至编码习惯，这一两年内涌现出了很多优秀的JavaScript MV*框架（称为MV*是为了兼顾基于不同设计模式和只实现了View和Model的框架），比如SammyJS、SpineJS、YUI3 Library等。每个框架都有自己的特点，这也是开发者和作者们热烈讨论的话题。

这里需要说明，JavaScript MV*框架与jQuery这类功能库有本质上的区别。JavaScript MV*框架只负责搭建SPA最基本的架构，本身并不会对DOM进行操作，也不关心界面及交互的浏览器兼容问题等。而jQuery这类功能库没有提供架构层的任何解决方案，因为这也不是它的职责。

BackboneJS

BackboneJS必须要依赖UnderscoreJS，并需要选择一个基础功能库和一个前端模板引擎来共同实现一个完整的SPA。

BackboneJS 体积很小，在 600 多行的代码中实现了 View、Model、Collection、Events、Router、History、Sync 等重要概念。为开发者提供了简单、清晰的组织结构。同时在开发方式及基本类库选择方面也足够灵活，深受微框架爱好者喜爱。

在官方文档中作者从来没有声明它是一个JavaScript MVC框架。这是因为它虽然基于JavaScript MVC实现了View与Model的分离，但并没有实现Controller这一层，而是用View加Router来完成类似Controller的职能。

选择BackboneJS的原因

豆瓣说是豆瓣广播的升级，虽然已经下线，但它是豆瓣第一个使用BackboneJS的产品。豆瓣阅读的阅读器是一款为阅读作品提供分页、精美版式及读者沙龙等功能的Web应用。

从产品需求看，两者都是一个标准完整的SPA。另外开发周期较短，学习成本低，有成功实例且文档完善，相对稳定的JavaScript MV*框架。BackboneJS实现了SPA中最重要概念的定义及其操作的最小集，不限制开发思路的同时保持代码的合理组织，灵活的自定义事件绑定，丰富实用的枚举函数及方便的API，以及在视图中声明事件处理函数。

能够满足产品需求，以及框架本身的优秀让我们最终选择以BackboneJS作为豆瓣两款产品的基础。但它足够轻量和灵活的背后也意味着需要开发者自己实现的东西会很多，这点在复杂且大型的应用中体现得比较明显。

Backbone的 backbone

图1列出了BackboneJS提供的所有核心对象，及其主要属性和方法。我们将结合实际项目详细介绍BackboneJS的核心API，并分析其实现原理。

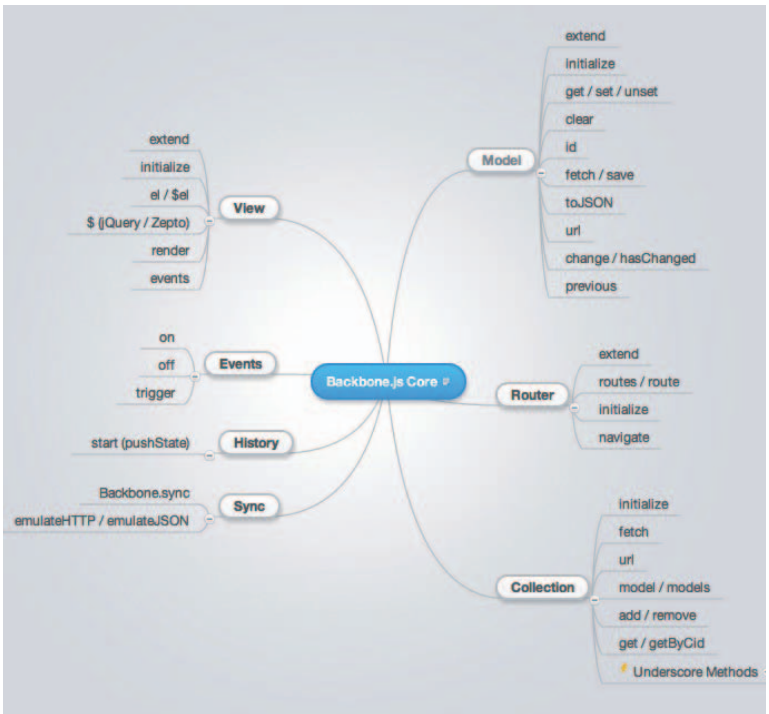


图1 BackboneJS的核心结构

Backbone.Events

BackboneJS是通过事件机制来保持数据与视图的一致的。

所有对象都可以通过对Backbone.Events做扩展来绑定自定义事件，事件在被绑定之前不需要声明，还可以携带参数。

```
var object = {};
_.extend(object, Backbone.Events);
object.on("alert", function(msg) {
      alert("Triggered " + msg);
});
object.trigger("alert", "an event");
```

BackboneJS已经将Model、Collection、Router、History及View的原型都与Events做了扩展，并在相关方法中触发了change/add/remove/sync/all等最常用的事件，开发者只需在视图中对数据（或集合）做事件绑定，定义相应的处理函数。

在实际开发中，除常用事件外，还会频繁使用自定义事件。例如在阅读器中为翻页视图绑定自定义事件，以防止用户连击。

Backbone.Model

Backbone中的Model是整个应用中最核心的部分。Model与界面的具体呈现及交互都无关，只涉及本身的定义，各种操作（创建、更新、删除、验证等）及存储。

豆瓣说中的广播、评论、用户，豆瓣阅读器中的作品、提醒等都是Model，可以用类似下面的方式定义。

```
var Comment = Backbone.Model.extend({
    defaults: {},
    initialize:function(attributes,
                        option) {},
    ...
    validate: function() {},
});
```

除了通过键值对来描述数据外，BackboneJS还提供了Model的最基本属性和常用方法。

Backbone.Collection

Backbone.Collection即Backbone.Model的集合。豆瓣说中的广播流、回复列表、关注列表，豆瓣阅读器中的作品列表、提醒列表等都是相应的Model集合。所以在定义数据集合时需要指定model属性值：

```
var Comments = Backbone.Collection.
extend({
       model: Comment,

       initialize: function(models,
                            options) {
          this.url = '/api/statuses/' +
                     options.id + '/
                     comments';
       },
       comparator: function(model) {
        return model.get('id');
       }
});
```

请注意，在使用collection.create()方法往集合中追加数据时，向服务器发出POST请求的地址并不是collection的url，而是collection.model的urlRoot。因为create()方法实际完成了三件事：初始化model，再向服务器端写数据，最后将返回的数据添加到集合中。

```
collection.create(attributes) =
new model(attributes) + model.save() +
collection.add(model)
```

另外，Backbone.Collection除了定义基本属性和方法，还可直接使用UnderscoreJS中28个灵活高效的迭代函数完成常用操作。

在这里还有必要简单介绍下UnderscoreJS，它同样是由Jeremy Ashkenas开发的一款JavaScript实用工具库，提供了类似Prototype功能的编程支持，但没有对JavaScript内置对象进行扩展。它提供了常用的对象判断，数据集合和数组操作，以及一个模板引擎。它不但是BackboneJS必须依赖的功能库，也经常被开发者单独放到项目中使用，很大程度上弥补了JavaScript本身对于数组操作接口的不足。

Backbone.View

在SPA中，大到页面，小到按钮、链接都可定义为视图对象。Backbone对View的定义很精简，其中基本功能库及模板引擎都由开发者自己选择。在自定义视图对象时通常要做以下几件事情。

1. 定义视图基本属性

view.el：由view的tagName、className、id和attributes等属性值决定的。如果没有特别声明，el会是一个空div。

view.template：定义视图中需要的模板文件。

2. 定义初始化方法

在实例化视图时，会先调用initialize()方法，为数据（集合）绑定事件处理函数、读取相应数据等。

```
var StatusItem = Backbone.View.extend({
    ...
    initialize: function() {
   this.collection.on('add', this.addOne,
```

```
                    this);
    this.collection.on('remove', this.
                    addAll, this);
    this.collection.fetch();
    ...
    },
    addOne: function() {},
    addAll: function() {}
});
```

3. 绑定DOM操作事件（view.events）

```
var StatusItem = Backbone.View.extend({
    ...
    events: {
        'click .btn-like'    :'like',
        'click .btn-reshare' :'reshare',
        'click .btn-delete'  : 'delete'
    },
    like: function() {},
    reshare: function(e) {},
    delete: function(e) {},
    ...
});
```

这里实际使用了jQuery或Zepto的事件绑定方法：bind或delegate。通过分析events的key和value，将key中的事件名称与DOM元素节点区分开，再用jQuery的delegate方法监听事件。

4. 定义视图渲染（view.render()）

Backbone.View的render方法定义是空，在自定义视图时，需要重新定义render方法，且在最后返回视图本身。

```
var StatusItem = Backbone.View.extend({
    template: $('#status-item-template').
              html(),
    render: function() {
        this.$el.html(Mustache.to_html
                  (this.template,
                   this.model.
                   toJSON()));
        return this;
    },
    ...
});
```

在实际开发中，我们会遇到一些方法属性相似的对象，这时我们就要合理地将其抽象出来，之后再对其进行扩展。在豆瓣说中，广播流（Stream）视图就是抽象出来的基本视图对象，在此基础上会扩展出其他页面中的广播流对象（HomeTimeline/UserTimeline/Topics）等。

Template

在View中还有一个重要概念：前端模板。这里简单介绍一下豆瓣说采用的前端模板引擎MustacheJS。MustacheJS相对于其他模板引擎有两大特点：

- 支持多种语言：除JavaScript外，还支持Ruby、Python、Go、CofferScript等22种其他语言；
- Logic-less：这个特点其实是好的方向，可以让数据更加纯粹，不带有任何逻辑。

目前有一些模板引擎是由MustacheJS扩展而来：HandlebarsJS和HoganJS。当然Mustache在性能方面不是很优秀，在一些古老的浏览器当中表现不好。期待作者对于Mustache2.0（http://writing.jan.io/mustache-2.0.html）的设想尽快实现。

Backbone.Router

从0.5.0版本开始，Backbone.Controller更名为Backbone.Router，配合View共同完成MV*中控制台的职能。定义应用路由方式如下：

```
var Shuo = Backbone.Router.extend({
    routes: {
      ''              : 'home',
      ':uid'          : 'profile',
      'search/:query' : 'search'
    },
    home: function() {},
    profile: function() {},
    search: function() {}
    ...
});
```

通过对Backbone.Router做扩展来自定义路由类，routes属性里创建了应用中路由键值对，并定义URL变化且匹配时要调用的相应函数，实现了每个“页面”都可以通过URL来定位和传播（Backbone.Router本身功能的实现更多的是调用了Backbone.History中的方法）。

Backbone.History

针对不同的浏览器，在解决浏览历史问题时有以下两种策略。

- Hash Change

利用window.location.hash属性可以生成一个新的URL，但不会发起新的请求。在支持onhashchange事件的浏览器中，我们就可以监听到URL的变动，然后再触发相应的处理函数，最终达到切换相应视图的目的。每次location.hash属

性改变，浏览器都会自动更新history中的记录。

■ HTML5 History API

HTML5为historyAPI引入了两个很实用的方法：pushState()和replaceState()。

pushState(stateObject, title, url)会将url添加到当前的浏览历史记录中，支持此方法的的浏览器会在网页“前进”或“后退”时触发popstate事件，且不会向服务器端发起新的请求，更不用去管这个url下是否存在真实的页面。它的效果只会影响history中的记录。Backbone.History就是依照上面的策略来完成SPA中浏览历史的管理及路由的映射的。

Backbone.sync

每当应用中的数据需要与服务端有交互时都会调用Backbone.sync，将method（CREAT/PUT/GET/DELETE）传入，再发起相应的网络请求。Backbone.sync本身其实是利用jQuery和Zepto的ajax方法发起RESTful json请求。当然，如果你的应用的数据存储策略有所不同，也可以根据自己的实际需求重新定义Backbone.sync对象。

其他Javascript MV*框架的比较

EmberJS

由SproutCore发展而来， 支持数据的可计算属性、双向绑定等特性，为Web应用开发提供了一整套完整的解决文案：将数据和事件直接绑定在模板里，且明确定义Model、View和Controller类。虽然没有指定前端模版引擎，但默认的HandlebarsJS提供了除MustacheJS基本语法外的自定义标签等功能。目前EmberJS的社区还不太强大，文档的内容与其提供的功能相比很不成比例，提高了学习成本，很多最佳实践也没有指明。

YUI3 Library

其思路与BackboneJS大致相同，主要提供了Controller（负责路由功能）、Model（数据的属性及基本操作，与数据存储层交互时触发的基本事件声明）、Model List（数据集合）及View（负责渲染视图及响应用户操作，监听数据变化）等组件，同时还有强大的数组迭代函数及应用开发中的各种插件支持。

SpineJS

是一个与BackboneJS最为接近的框架，体积也相当精小，但不同的是View层就是单纯的指模板文件，而Controller实际跟BackboneJS中的View相同。其作者出版的《Javascript Web Applications》一书中有更为全面和详细的介绍。

Javascript MVC

SpineJS除提供了$.Class、$.Model、$.Controller和$.View等基本类，还包括了测试框架、模块化管理工具，以及产品发布和文档等实用工具。目前还对Less和CoffeeScript做了支持。自2008年发布以来已经慢慢发展成一整套完善的JavaScript MVC框架。

小结

利用以上BackboneJS提供的最基本的对象及操作，再加上基本功能库和前端模板，我们便可快速搭建一个完整标准的SPA了。但在越来越复杂的应用开发中，除了功能开发外，产品的发布及测试维护也是我们要考虑的重要事情。例如可以利用RequireJS对代码进行模块化管理，按需加载，并在发布环节对JavaScript文件进行压缩。另外，通过JasmineJS对代码进行单元测试以保证其质量。完善开文档也是团队合作中相当重要的部分。



吕婷

豆瓣前端工程师。Web标准化交流会工作人员。豆瓣UID：imdonkey

责任编辑：卢鹄翔（ludx@csdn.net）

新书上架

本月新书



创新天才

作者：Peter Fisk

译者：张冠南 相娜 赵俐

本书从“设计模式”的角度来研讨和归纳创新的方法，以及走向商业成功的方方面面，介绍了创新的数十条轨迹，并为进行创新提供了商业蓝图。



任天堂传奇

作者：Daniel Sloan

译者：张玳

任天堂，经过一百多年的起起伏伏却依然能傲视群雄，什么样的战略和战术在发挥作用？合作、竞争这永恒的主题在商场上又该得到怎样的诠释？让任天堂和它的竞争对手告诉我们答案吧。



差错：软件错误的致命影响

作者：Jeff Papows

译者：邝宇恒 陆子陶

本书全面分析了软件错误问题为何愈演愈烈的原因，为各类公司的决策者提供了切实可行的软件管理技巧和方法，从而降低并预防软件错误对公司的影响。



JavaScript语言精髓与编程实践（第2版）

作者：周爱民

本书详细讲述JavaScript作为一种混合式语言的各方面特性，包括过程式、面向对象、函数式和动态语言特性等。意在分解出这些语言原子，并重现将它们混合在一起的过程与方法。



HTML5移动开发即学即用

作者：王志刚 等

本书逐一剖析HTML5标准中包含的最新技术，详细介绍了HTML5新标准中提供的各种API，同时附上了相关的应用实例，且大部分可以直接应用于自己的HTML5程序中。

推荐图书



软件之道：软件开发争议问题剖析

作者：Andy Oram Greg Wilson

译者：鲍央舟 张玳 沈欢星

出版社：人民邮电出版社

在软件行业中，如何选择技术和流程、如何合理利用工具、什么才是最佳实践，相关的讨论和探索每时每刻都在进行。然而尽管经历了长时间的争论和不停息的论证，行业普遍认同的标准软件开发方法却似乎并未形成。不管是因为软件行业是新兴产业，经验积累和知识沉淀还远远不够，还是另有缘由，现阶段的软件开发，确实没有权威方法论的存在。

某些编程人员的工作成效果真是他人十倍之多？测试驱动开发果真能帮助更快、更好地开发代码？软件的Bug数量果真可以利用代码度量进行预测？设计模式果真有助于构建更好的应用程序？人员个性会对结对编程产生何种影响？地理位置的距离和公司职位的差距，究竟何者影响更大？

以上任何一个问题都有可能引发一场激烈的口水仗。然而面对这些看似主观的问题，孰是孰非又十分难以界定。本书的意义就在于，试图找到可靠的定性和定量证据来回答这些问题。这些已不再是没有事实依据的论点，是时候应用科学的方法和收集确凿的证据，对其做出公平公正的推断了。

本书集合了几十位软件工程领域顶尖研究人员的实证研究，通过呈现他们长达几年甚至几十年的研究成果，揭示了软件开发社区普遍存在的一些确凿事实和虚构之事。书中探讨了更有效的编程语言，对比了软件开发人员之间的效率差异，验证了康威定理，并反思了软件行业的最新模式。通过各种实证论文引导和帮助软件行业从业人士分析现有的实证，并去探索和寻找切合自己的方法。

书中一个重要的观点是，那些我们所熟知的技术、流程、工具、实践所产生的作用是因人而异的。每个人都需要从自己的实际角度出发，判断证据的可信度和适用度。因为软件开发的复杂性决定了，对别人适用的证据并不一定对你适用。

不应该对业界的一些热点话题盲目听信或者盲目反对。如本书第2章最后一小节中说到的，作为一个软件工程师，每个人都不应该放弃判断现有证据的资格和能力。分析现有证据中的数据，并结合自己的实际情况做出相应的判断，是每个专业工作者的责任。盲目的跟风或者盲目的批判都不是专业的表现。

同时，本书也对业界的研究者提出了更高的要求。在还不算成熟的软件行业中，我们应该如何向更成熟的行业（如医学）学习，从而把已有的研究有效地聚集起来，形成对整个行业都有用的数据库。在此基础之上，从业者们可以更好地找到适应特定环境的证据，也能更容易地做出判断。

本书将帮助读者拓宽视野，更好地选择适合的工具和技术，并最终成为一名更加优秀的软件行业从业人员。适合所有软件开发人员和研究人员阅读。



社交网站的数据挖掘与分析
作者：Matthew A. Russell
译者：师蓉
出版社：机械工业出版社

数据挖掘，英文为Data mining，又译为资料采矿、资料探勘，它是数据库知识发现（Knowledge-Discovery in Databases，KDD）中的一个步骤。数据挖掘一般是指从大量的资料中自动搜索隐藏于其中的有着特殊关联性的信息的过程。通常与计算机科学有关，并通过统计、在线分析处理、情报检索、机器学习、专家系统（依靠过去的经验法则）和模式识别等诸多方法来实现。

近年来，Twitter、Facebook、LinkedIn这些社交网络迅猛发展，逐渐改变了人们的生活和交流方式。它传播的信息已成为人们浏览互联网的重要内容。

然而，互联网中充斥着大量的信息，且有价值的信息和没有价值的信息混杂在一起。这是个令人困扰的问题。因此，如何有效地利用社交媒体，如何读取数据，如何从原始数据中提取出有价值的信息便成为亟待解决的问题。

这便是本书的价值所在。对于任何想要涉足社交数据挖掘分析的人来说，本书都是你的必备参考书。它不仅介绍了挖掘数据、分析数据的技术，还提供了对数据管理工具以及可视化方面话题的讨论。此外，为了能让读者充分了解这些技术，Matthew还提供了简单易懂的示例和非常有价值的建议。

本书的每一章都将社交网站与数据挖掘、分析和可视化技术的内容组织在一起，来回答以下问题：谁与谁相识？他们共同的朋友是谁？某人与其他人多久联系一次？人与人之间的交流在多大程度上是对称的？网络中最沉默/最健谈的人是谁？网络中最具影响力/人气最高的人是谁？人们正在聊什么（而且它有趣吗）？

要回答这些问题，通常都会涉及两人或更多的人，并且需要找出他们之间存在关系的上下文。回答这些问题所涉及的工作只是开始，更复杂的分析过程还在后面，但你必须找到切入点，因为社交网络API和开源工具包都很容易掌握。

如果你有编程基础，并且对通过社交网络进行数据挖掘和分析来洞察身边的机会感兴趣的话，你就选对书了。在短短几页介绍之后，就可以开始动手编写代码。

然而，从零开始构建自己的自然语言处理器、探究可视化图库的典型用法，以及任何与构建相关的最新技术这类活动，都不属于本书的范围。如果读者想了解以上任何内容而购买本书的话，一定会很失望。且并不能仅仅因为“在区区几百页中进行文本分析或记录匹配既不真实，也无法体现我们的最佳技术”，就认为本书无法让你找到疑难问题的合理解决方案，无法将这些方案应用于社交网站，在此过程中并无乐趣可言。

当然，这也并不妨碍你在这些诱人的研究领域中培养积极的兴趣。

全球排行榜

Amazon

01 Turing's Cathedral: The Origins of the Digital Universe
02 HTML and CSS: Design and Build Websites
03 Engineering Long-Lasting Software
04 Cracking the Coding Interview: 150 Programming Questions and Solutions
05 Liars and Outliers
06 JavaScript & jQuery: The Missing Manual
07 Objective-C Programming: The Big Nerd Ranch Guide
08 Programming in Objective-C, 4th Edition
09 Inside Apple: How America's Most Admired--and Secretive--Company Really Works
10 Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition

天珑书局（中国台湾）

01 版本控制使用Git
02 Android 初学特训班
03 App 程序设计入门—iPhone、iPad，2/e
04 一定要学会的HTML5 + CSS3网页设计实作应用
05 Google Android SDK 开发范例大全，3/e
06 Android 技术内幕：探索Android核心原理与系统开发
07 系统分析与设计
08 Google Android应用程序开发实战，3/e
09 Android 4.X 手机/平板计算机程序设计入门、应用到精通，2/e
10 Eclipse完全攻略：从基础JAVA到PDE外挂开发

第二书店

01 计算机操作系统（第3版）
02 研究之美（英汉对照）
03 社交网站的数据挖掘与分析
04 深入学习MongoDB
05 高性能Linux服务器构建实战:运维监控、性能调优与集群应用
06 Java编程思想（第4版）
07 Spring 3.x 企业应用开发实战
08 C程序设计语言（第2版·新版）
09 Java并发编程实战
10 深入剖析Tomcat

Mac OS X 背后的故事（十一）

半导体的丰收（下）

文 / 王越

随着CPU与GPU合并成技术发展的趋势，苹果开发出了OpenCL框架，能够进行高速并行处理的能力使OpenCL成为了业界标准，被广泛应用。

最近几年，GPU的发展吸引了很多来自科学计算界人士的目光。GPU有稳定的市场推动力——公众喜闻乐见的电子游戏产生了源源不断的升级GPU的需求——因此比CPU的更新步伐更快。从技术上讲，GPU本身就是多核架构，高端显卡往往有五百多个核心，即使低端的集成GPU也有二三十个核心，所以能够通过并行来高效处理成千上万的线程。同时，对于科学技算中的浮点计算，GPU往往通过硬件加速使其效率比传统CPU更高，因为图形渲染等工作基本都是浮点计算。

GPGPU浮出水面

早期的GPU只能执行固定的程序，而不开放给程序员编程。随着时代的发展，图像处理有时需要对着色器进行编程以实现一些特效，因此需要程序员可以使用GPU的汇编语言写简单的着色程序。这自然对程序员要求过高，所以一些高阶的着色语言又被GPU厂商开发出来。比如微软和NVIDIA共同开发的Cg语言，就能为顶点和像素编写专门的着色程序。这类技术虽然面向图形渲染工作者，却吸引了一小簇科学计算研究者的兴趣。以计算流体力学为例，它是用纳维斯托克斯方程【注：把牛顿第二定律和质量守恒应用到流体后，所得到的偏微分方程】来求解流体力学问题的一种算法，广泛用于天气预报、F1方程式赛车设计等工程领域。同时，对于电影制片特效，计算流体力学也是最基本的用来模拟流体流动特放的算法，皮克斯动画工作室的《寻找尼莫》中的海洋流动和水花等，都是使用纳维斯托克斯方程来模拟的。

首先，对于一个几何空间进行网格化，每个网格中的流体，都可以列出纳维斯托克斯方程，把这些方程联立起来进行求解，即可得到各点的温度、压力、湿度、速度等流体信息。整个求解过程可以高度并行，因为每个网格的控制方程是完全一样的；同时也牵涉大量的浮点运算。但Cg这类语言并非面向普通的计算，其变量都是颜色、顶点、像素等图形学专用变量。来自北卡罗莱那大学教堂山分校的Mark Harris突发奇想：可以把流体力学中每个网格的速度、压力等变量，存成RGBA颜色后让Cg去处理，所以他在《GPU Gems》中著名的一章，公布了使用Cg来高速实现计算流体力学运算的成果，吸引了大量计算界的目光。然而，这种编程模式对科技工作者来说很不友好，因为这要求一个学力学的、学生物的、学化学的学生，先要明白复杂的GPU渲染原理，了解图形学中材质、顶点、合成、像素、光栅化、光线跟踪等深奥的理论，才能编写他们专业相关的GPU程序。

GPU生产厂商洞察到了GPU高速并行浮点数运算的潜力，所以GPGPU（General Purposed Graphics Processing Unit）概念终于浮出水面。一方面GPU设计一代比一代可编程化，另一方面各公司也在加紧研制新一代GPU编程语言。新一代的语言对比Cg，去掉了对于渲染相关的知识要求，独立于图形学之外，是纯粹的普通语言，比如变量不再是像素、顶点、面等类型，而是C/C++语言开发者喜闻乐见的浮点数组、整形数组等。这一时期为代表的语言，主要是CUDA（Compute Unified Device Architecture）。CUDA是NVIDIA在2007年公布的一项面对科学计算工作者的编程框架。通过该技

术，使用者可利用NVIDIA的GeForce 8以后的GPU和较新的Quadro GPU进行高性能编程。用户先编写一个特殊的C++代码文件，扩展名为cu，文件中需要申明创建的变量、GPU计算核心（kernel）以及使用给定的编程接口来实现变量在CPU和GPU中的传送。然后通过NVIDIA自家的编译器编译这个代码，链接到NVIDIA自家的库上，即可把该运算核心编译为GPU汇编语句扔到特定型号的GPU上高度执行。其他厂家也紧随其后，比如AMD为ATI生产的GPU卡提供了一个类似的框架叫Stream SDK。而微软更是趁Vista和Win7推出了DirectCompute，作为旗下DirectX技术的一部分。

CUDA并不完美

对科学工作者来说，CUDA比Cg友好太多。使用CUDA加速流体力学运算相关的论文更是雨后春笋般涌现。然而不久后，我发现它存在许多问题。

首先，对初学者来说，CUDA编程模式很容易学混。因为一个GPU数组和一个CPU数组在CUDA中的表述都是同样的C指针，但对于GPU数组和CPU数组，CUDA的处理模式完全不同，CPU数组使用常规的malloc来初始化，而GPU数组得使用CUDA提供的malloc。所以程序写着写着，就忘了一个变量到底是给CPU用的还是给GPU用的，这无疑增加了学习难度。同时，CUDA对C/C++语言进行了一系列扩展，这不但意味着写的程序不再具有C/C++那样良好的可移植性，而且这种计算核心和传统C程序混写的编程语言很不美观。

其次，CUDA这类语言的实现各自为政。如果你写了一个CUDA程序，就意味着这个代码只能运行在NVIDIA的显卡上。如果想使用ATI的显卡呢？没门，请用ATI Stream SDK重写。

再次，CUDA是在编译时就静态产生GPU代码的，所以只能产生特定的GPU代码。如果你发布了一个CUDA程序，它仅对某几种NVIDIA显卡进行特定的代码优化。如果NVIDIA自家出了一种新显卡，很抱歉，哪怕新显卡可能兼容老显卡的汇编指令而你的程序恰巧可以在新显卡上跑起来，你也无法发挥新显卡的所有特性。必须用针对新显卡的编译器重新编译源代码，才能够保证程序在新显卡上高效执行。

最后，CUDA这类语言仅能产生高效的GPU代码，而无法产生CPU代码，即：写完的代码只能跑在GPU上，在CPU上只能“模拟执行”，仅供调试用。所以在一台不具备给定GPU的机器上，无法高效运行CUDA程序。同样，如果你有一个性能很强的工作站，那么你的CPU毫无用处——CUDA不可能分配一部分任务给CPU完成。

另外还有未来计算机架构的不确定性。当时，GPU越来越一般化，可以跑多种数值计算程序，而CPU随着多核成为主流也越来越像GPU。所以很多厂家在考虑CPU和GPU合并的可能性。

当时轰动一时的热门事件，是CPU厂商AMD买下了GPU厂商ATI，来开发下一代处理器AMD Fusion，把GPU和CPU合并到一起。Intel自然不甘示弱，做出了Nehalem平台，在该平台上，CPU和集成GPU处于同一个包装中，外界一度猜测这样可使合并后的CPU具有图形处理工能，从而用户购置计算机就不用再考虑配一块GPU了。

更强大的是，当时Intel还公布了Larrabee计划，让GPU支援x86指令，使得一个常规的x86平台的程序不需要修改和重新编译便可在GPU上运行。

虽然事实和这些预期有稍许出入，但当时的技术趋势是：在将来可能出现一种新的合并GPU/CPU的技术，能够并行高速地运行一般的计算机程序，而面对这样新的可能的平台，我们如何准备？

OpenCL诞生

OpenCL则是苹果为这个新局面画下的蓝图。这项技术初期全称为Open Computing Library（如果留意苹果早期宣传广告的话），后改名为Open Computing Language。这项技术从本质上来说，和CUDA并没有太多的两样，但由于苹果在借鉴他人技术并把他人技术改得更棒这一点上是出了名的，所以OpenCL很好地解决了以上所有问题。

下面简单介绍一下这个框架。OpenCL技术的结构十分清晰，对程序员来说，它是一个Mac OS X的Framework，定义了两套标准，一套是一个C语言的编程界面（API），使得开发者创建、拷贝、

回收GPU使用的对象，同时也包含检测处理器、为该处理器编译并调用核心程序（kernel）相关的接口；另一套是OpenCL核心程序语言的定义，是一套基于C99发展而来的语言。

例如我们有两个大数组，1024维的a和1024维的b（当然，1024不算大，OpenCL往往用来处理十万、百万数量级的任务），我们把两个数组对应的元素加和，结果是一个1024维的数组c。C程序员很容易能写出下面的程序：

```
for (int i = 0; i < 1024; i++)
c[i]=a[i]+b[i];
```

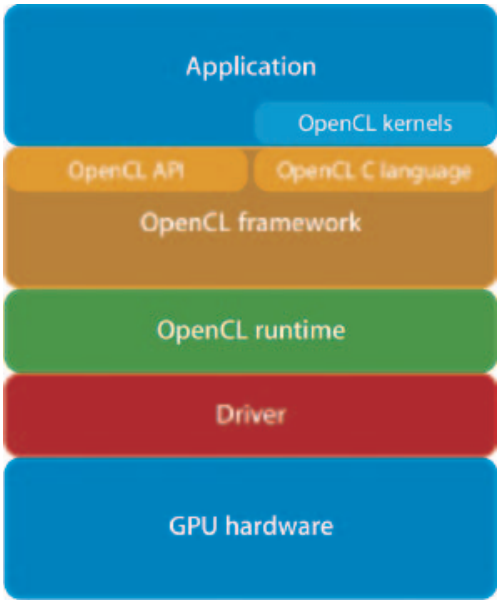
OpenCL的核心程序，则是取每个独立的可并行的循环分支，即上面程序中的 c[i]=a[i]+b[i]。所以核心程序大概是下面这样：

```
__kernel add(float *a, float *b, float *c){
int i = get_global_id(0);
c[i]=a[i]+b[i];}
```

其中，get_global_id()函数可以返回当前函数是全局中的第几个元素。把该程序保存为add.cl，就是一个OpenCL的核心程序，为C99语言的一个子集。

使用OpenCL的API就能调用这个核心程序。每个OpenCL程序基本上是模式化地照搬下面流程：

1. 探测硬件（用clGetDeviceIDs函数护取计算设备（可以指定使用GPU或是CPU），用clCreateContext函数来新建一个上下文（context），用clCreateCommandQueue函数针对设备和上下文新建一个命令队列）；



OpenCL架构

2. 编译核心（读入add.cl，用clCreateProgram-WithSource和clBuildProgram以及clCreateKernel来编译读进来的字符串，产生一个核心程序）；
3. 写入数组（用clCreateBuffer创建a、b、c三个内存对象，用clEnqueueWriteBuffer把C数组写到内存对象中）；
4. 运行核心（把内存对象作为核心程序函数的输入参数执行这个核心，程序会并发为1024个线程，每个线程执行一次相应的加法运算）；
5. 读出结果（用clEnqueueReadBuffer读取c内存对向，写为C的数组）；
6. 回收内存。

OpenCL之美

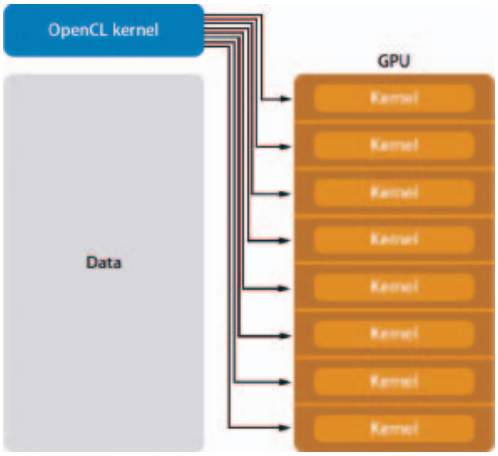
让我们逐条来看前面那些问题是如何被解决的。

首先，OpenCL Framework由C API和OpenCL语言组成，泾渭分明，所有的GPU变量在C API中，都是内存对象的形式出现，有别于C自建的数组。因此，你永远不会搞混两者。同理，OpenCL核心程序是独立在C源程序之外的，不仅美观，也能保证你的C程序能被所有C编译器编译，因为调用OpenCL库和调用其他C的函数库没有任何不同。

其次，苹果开发出OpenCL后，觉得该技术甚好，索性联合AMD、ARM、ATI、TI、Intel、IBM、Nokia等公司，把它做成一个由Khronos组织主持的开放标准。不管电脑上用的显卡是ATI的还是NVIDIA的，OpenCL都能像OpenGL那样在你的设备上无缝运行。事实上，OpenCL已同OpenAL和OpenGL一样，成为Khronos Group旗下的三大业界标准。

再次，CUDA是在编译时就静态产生GPU代码的，所以只能产生特定的GPU代码。而OpenCL的核心程序（kernel）是在运行时被编译成GPU指令的。由于kernel所用的OpenCL语言，仅是C99的一个子集，所以负责编译这个程序的是OpenCL运行库自带的LLVM-Clang。这样做的好处是明显的，举例来说，如果用户有一堆OpenCL的程序，比如苹果最新的Final Cut Pro X，如果某一天硬件厂商发布了一个全新的GPU架构，那么用户安装显卡后，只要下载或更新相关的驱动程序和运行库即可，而不需要再求软件厂商发布一个新版本的Final

Cut Pro X。因为OpenCL在运行时，会根据显卡厂商提供的驱动和新运行库自动优化程序到特定架构上。所以，程序兼容性问题也被圆满解决。



OpenCL的核心程序是在运行时被编译成GPU指令的

最后，由于OpenCL是个开放标准，也支持CPU和其他任何计算设备，比如数字信号处理芯片（DSPs）和各种专门的处理器架构。所以只要有相关的驱动和运行库，OpenCL程序可以高效地并行运行在任何架构的运算设备上。由于OpenCL和GCD的编程模式是一样的，因此当OpenCL程序在CPU上执行时，是跑在GCD队列上的。

OpenCL因其高速并行处理的能力被广泛应用，苹果也是OpenCL的主要用户之一。上文提到的Final Cut Pro X就是使用GCD和OpenCL进行大量并行的流媒体处理。在老版本Final Cut中，每当用户执行一次流媒体操作，都会弹出一个进度条来告诉用户剩余的处理时间，而Final Cut Pro X优化后的速度是如此实时，以至于这个进度条被去除了。Mac OS X许多底层库也使用OpenCL重写，如Core Image，本身也是一个GPU加速库，使用OpenCL后，相比原来，依然获得了非常可观的性能提升。

Snow Leopard的发布标志着第一个OpenCL框架的完整实现，OpenCL成为业界标准后，AMD抛弃了原先的策略，投入开放标准的怀抱，一连放出了几个测试版本的集成OpenCL的ATI Stream SDK，并在2009年年底发布了稳定版，2011年8月8日宣布废除原先的Close to Metal相关技术。NVIDIA也是早早地在CUDA SDK中加入了OpenCL相关的库。CUDA越来越不被看好，所以NVIDIA索性把CUDA发布为一个开源项目，并把CUDA架构在LLVM之上。这和OpenCL近几年的走强有很大关系。

开发者的瓶颈

目前看来，OpenCL虽然解决了上面的所有问题且且速度飞快，但对普通程序员来说，依然是非常底层的技术。而且由于硬件的限制（显卡不支持指针运算），很多C的标准并未在OpenCL中出现，写链表还需要用整数去模拟地址。程序员需要手动管理内存，处理底层的核心调用以及数据读写。而显卡厂商也大多不愿公开GPU的技术细节，因此不像CPU程序很容易通过汇编指令分析计算机底层干了什么，显卡对于开发者纯粹是个黑盒，把整个问题分成多少个线程并发也没有一个规律可循，有可能不起眼的改动会使程序运行瞬间变快或变慢数十倍，开发者也不知道其中的原因，只能凭经验操作。而且由于不存在良好的调试工具，所以很难改正程序的错误。

显卡作为系统最为重要的共享资源之一，不像现代操作系统那样提供内存保护机制，因此一个用户OpenCL程序的错误很容易导致整个计算机崩溃，所以经常是程序跑一遍后发现操作系统挂了，重启后发现了一个可能的错误，改完后编译运行，操作系统又挂了。我用OpenCL编写科学计算程序时，大量时间是在重启电脑而不是写程序。这些问题仍然阻碍着OpenCL被广泛采纳，不过，在科学计算界，已经涌现出了越来越多相关的论文和技术，相信在不久的将来，情况会有所改观。

结语

当写完这篇技术长文时，天色已晚，走出教室，和ENIAC擦肩而过。ENIAC的出现激励了之后一次次的处理器革命。2009年发布的Snow Leopard可能在整个Mac OS X发行版历史中不算最出彩，却是对于半导体集成电路革命的一次重大收获。



王越

美国宾夕法尼亚大学计算机系研究生，中国著名TeX开发者，非著名OpenFOAM开发者。

责任编辑：陈博（chenbo@csdn.net）

从工程师到管理者：自然而然的转型

雅虎北京全球研发中心高级架构师梁启鸿专访

记者 / 董世晓

梁启鸿，从美国哥伦比亚大学硕士毕业至今，先后担任IBM T. J. Watson实验室Senior Associate Programmer、SUN大中华区客户解决方案部门首席架构师、创业公司CTO、雅虎北京全球研发中心高级架构师，经历可谓丰富，个人收获也是颇多。近日，《程序员》与他直面交流，分享他的程序人生。

不凡职业路

《程序员》：大学毕业后，您第一份工作是在IBM T. J. Watson Center，当时具体做些什么？

梁启鸿：虽然我的数学并不是很好，但我所在的那个小组切切实实是搞数学的，研究内容是知识表示（knowledge representation，KR）。在实验室这个听起来远离工业界的地方，我负责的却是技术实现，而非核心算法研究，因此更应该称作是“工程师”，而不是“研究员”。

当时，也就是1996年，Java刚推出不久，我就对Java产生了兴趣，并将它引入到我们实验室，用来做面向对象设计、软件开发等，例如KR系统的软件实现。

《程序员》：是不是因为这段Java的经历，让您后来去了SUN公司？

梁启鸿：我一直都对Java技术情有独钟，因此在离开IBM之后，在华尔街的多家投资银行包括摩根士坦利、摩根大通等都从事Java方面的技术工作，后来则去了美林证券，与SUN在纽约的全球第一个Java Center合作，用Java技术实现证券交易系统等金融相关的系统。

在那之后，我正式加入SUN，担任大中华区客户解决方案部门首席架构师，工作还是与摩根大通、美林证券、标准普尔等金融机构打交道。那时它们已经是SUN的客户，我们为其提供专业咨询服务。

《程序员》：后来为什么又想到创业呢？

梁启鸿：我始终觉得，做IT的人机会允许的话要有一次创业经历，姑且不论结果怎样，即使失败的经验也不是其他途径所能获得的。

身处大公司的人不能想怎样做就怎样做，会有很多掣肘，纵有千般想法、万般理念，也不一定有机会去实现。有时是商业环境造成的，因为大公司有不同的产品研发优先级，你的所想未必会与公司相匹配；有时是部门之间人际交往的复杂性造成的，因为大公司的部门和精英众多，你可能只是众多优秀员工中的一位而已。

此外，在大公司中，你可能不一定有机会参与全新的项目，因为它肯定会存在一些历史遗留问题，在技术成长过程中，回首看去，当时的很多设计并不合理，但由于它已经在被使用，没有办法去改造，所以你就不得不去适应它。而由于技术人员的天性使他更乐于享受原创，会对接手他人的代码有抗拒性，从而心情不爽。

而如果去创业的话，不管好坏结果都将是由自



梁启鸿职业经历丰富，感想颇多

己承担，个人拥有完全独立发挥的舞台。在创业的过程中，你所有的灵感和想法，即使再新、再“离谱”，也都能够让你有机会持续修改、扩展和提升。你的每一款产品都像自己的Baby一样，倾注了自己的感情，让你倍感珍惜。

因此，我觉得只有在创业公司里面才能真正享受到原创的成就感，完完全全按照自己或者团队的思路去做产品。

《程序员》：当时创业是做什么产品？

梁启鸿：我们做的是一个游戏平台。之所以选择这个方向，在于游戏平台的技术理念与金融业的需求有诸多相似的地方——都强调实时性、高并发、事务一致性。

为了能够做好这个平台，我们的大部分技术都是按自己的技术理念自己从底层做起，因为如果引入太多第三方技术的话，会让你不得不去适应它、改造它。当然，我们也采用了一些开源技术以及一些底层的库和框架，但总体架构上还是基于Java自己搭建的。我们也是最早开发Facebook上游戏产品的公司之一。

《程序员》：既然创业好处这么多，为何又加入雅虎呢？

梁启鸿：对这个问题，我自己也做过总结，可以说有两个方面。

一方面，雅虎的用户规模是我之前的工作经历中所没有机会接触到的，它的每个系统在做架构设计时，就需要考虑到支持全球用户在跨大陆数据中心的部署问题。不同用户规模带来技术挑战的维度差别很大，会导致一个完全不同的技术方案，是在一个比较小的范围里面所没有办法理解的。对我来说，这是一次难得的再次挑战自我、积累阅历的好机会。

另一方面，雅虎的工作文化是我比较喜欢的类型，保留了很多硅谷的特点。与传统IT企业相比，它灵活、不呆板，人与人之间单纯、友善，而且相对来说整个公司员工的平均技术素质比较高，可以互相学习和促进，让你真正体验到“技术无止境”。

《程序员》：您前面提到了在大公司的想法由于各方面的因素难以实现，那雅虎不也是一家大公司，也同样会遇到类似的问题吗？

梁启鸿：说实话，这个问题比较复杂。

首先，这与公司的文化有关系，看它鼓励创新的程度、给你多大的自由度。我觉得雅虎在这方面做得非常好。

其次是个人的做事方式，如果只是一心想着做

技术，而没有把技术与公司的商业兴趣结合在一起，只是觉得技术很酷、好玩，在任何公司都是要碰壁的。

再次，很多情况下，技术的推广一般会涉及到多个部门，要说服这些部门愿意成为使用者或者合作者，也不再是一个纯技术问题，一方面要让他们清楚这项技术的好处，能够有效帮助他们解决什么问题；另一方面要让他们能够成为你的同盟者，共同推动技术的推广。

弄潮移动互联网

《程序员》：也就是说，要找到最适合做的那个点，而不盲目为了技术而技术。那您现在在做哪些方面的工作？

梁启鸿：目前，我们主要在做一个移动平台。现在全球的移动界都在讲HTML5，所以我们也聚集了很大的火力在这上面。在Tablet和Smartphone日益强大的今天，网站和应用运行在移动终端是一个大趋势，但不应该是简单的移植，而是要加入创新的想法。我们的移动平台，目标是基于雅虎自己的云计算技术，建立一个solution stack，在多个层面为开发人员提供快速创新的方便。我们有一个理念上比较新颖独特的开发框架，有一整套的开发工具，有一个跨移动设备的HTML5/Native应用容器（container/runtime），配备一系列云端基础服务；我们还不断研究和增加新的针对移动类应用的增值服务，形成一个丰富的技术生态环境，通过API为各种创新应用提供方便。部分技术也许会在不久的将来开源。

补充说明一下，准确地说，所谓“移动平台”也许已经是一种过时的说法。在Tim Cook所说的“后PC时代”，移动已经成为个人计算甚至企业计算的基本属性。所以我们的技术考虑，是一个桌面Web和移动Web一体化的方向。首先，跨平台是一个举足轻重、优先考虑的点；其次，开发周期要尽量缩短，从而使想法变成现实并迅速地到达目标用户群体；再次，可伸缩性要好。业界的趋势是通过云端服务来进行协作，在云端建立好后端的服务，让开发人员轻松地重用这些现有的资源。

《程序员》：HTML5虽好，但依然不够成熟，您遇到的HTML5的最大问题是什么？

梁启鸿：毫无疑问，HTML5的出现，使得开发者可以轻松构建更丰富、互动性更强的应用，但目前还存在标准不够完善等问题。而我所关注的，也是业界一直存在的争论——怎样优化HTML5的性能。很多在Native上展现良好的应用，用HTML5来做的话，效果并不理想。为了解决这个问题，有些人提出了一种综合模式的解决方案，称为Hybrid App，使开发出的应用一部分是HTML5的，一部分是Native的。

但目前来看，对于很多技术人员来说，能否用HTML5来实现最“地道”的iOS或者Android界面效果、给用户最Native的操作感觉，是最重要的。这也可能是目前最受关注的问题。

转型是一个自然而然的过程

《程序员》：从工程师到管理者这一步步走来，您是怎样实现角色转换的？

梁启鸿：我感觉这是一个自然而然的过程。每位工程师的性格都不相同，有些人喜欢针对个别领域钻研得很深；有些人会看得全景一些，具备综合素质而不是成为某个领域的专家；还有一些人比较乐于去做带领者。

每当我看到同事代码不太规范、人员太多带来沟通不畅等问题出现时，总会积极去帮助团队尽量减少这些问题带来的影响。我习惯于将自己的心得整理和升华，变成一个团队“标准”，说服大家去遵循。虽说这个所谓的标准并不一定最好，但总得有人来做这件事情，然后，在执行的过程中根据大家的意见不断完善。在工作历程中，如果你总是执着地希望团队成员都保持技术“洁癖”，和你有相容的思维和习惯，那么你就会持续将自己的想法和思路去影响团队。个人觉得愿意冒险创新和对技术理念执着的人，比较容易影响他人。可以说，有影响力的人容易成为team leader。

硅谷创业之父
Paul Graham

文 / 阮一峰



2012年2月22日的《旧金山纪事报》，这样介绍Paul Graham：“他资助了如此之多的创业公司，以至于走在旧金山街头，一定会遇到某个他给过钱的人。”

这绝不是夸张。出生于1964年的Paul Graham，目前是硅谷的核心人物，在美国互联网界几乎无人不晓，硅谷到处都是与他有关的创业公司。他的创业理念和提倡的创业范式，已成为业界的共识。世界上第一个互联网应用软件Viaweb，就是由他在1995年开发的。它允许用户通过浏览器操作服务器，最终制作出一个网店。这在当时是革命性的想法：不用任何本地安装，只通过浏览器，就能使用软件。

1998年，Yahoo!以4900万美元的价格，购买了Viaweb。这不仅让Paul Graham获得了财富，更让他确信，一个软件开发的新时代已经来临，互联网将彻底改变软件工业，谁先掌握了新的观念，谁就能脱颖而出。他将自己的思考写成一系列文章，2004年结集为《黑客与画家》一书。虽然此书写于互联网时代的早期，但由于观点具有高度的前瞻性和启发性，以至于将近10年后还没有过时，每年依然有大量读者阅读，已成为互联网行业公认的经典书籍。

互联网创业是Paul Graham思想的核心领域。2005年，他决定将自己的想法投入实践，成立了孵化器Y Combinator（简称YC），帮助那些有技术、有创意、但不知怎么做的年轻程序员，尽快创业成功。

他将创业过程总结为一条曲线，认为大多数创业公司成立之后，会经历一个长期的低谷阶段，只有少数公司能够坚持到迎来曙光的那一刻。YC的使命，就是帮助和引导创业者顺利起步，以最小的代价、最短的时间度过低潮期，顺利进入后面的稳定上升期。

YC主要投资于互联网和移动应用项目，每年举办春季和冬季两次训练营，每次有30个左右的创业团队参加。YC向这些团队提供2万美元左右的种子资金，换取2%～10%的股份。创业者入选后，必须在硅谷集中开发三个月，最后在“展示日”（Demo Day）上拿出产品原型，以期获得风投公司的青睐，拿到风险投资。YC向创业者提供的，不仅仅是资金，还包括悉心的指导。此外，YC还提供丰富的人脉资源和关系网，以及创业者之间的互相交流和激励的环境。Paul Graham曾经这样说：“你知道YC最大的优势是什么吗？它让你不必加入大公司就能享受到大公司的好处。创业的困难在于，即使你各方面都很优秀，但你没有基本的资源，就很难成功。创业很孤独，没人给你出主意，所有事情都必须自己做。如果你为大公司工作，也许各方面都受到制约，但无论遇到什么问题，可能都能找到1000个人，一起来解决。现在，YC也能让你找到1000个人，一起来解决问题，并且没有大公司的任何束缚。这种关系网络以前是不存在的，可YC让这变成了现实。”

正因如此，YC资助的创业公司成功率非常高，著名的Dropbox、Reddit、Airbnb、Disqus都曾是YC的入选者。Paul Graham的影响力，伴随这些公司一起，在硅谷不断扩张，跻身为主流。

功成名就并没有让Paul Graham停止脚步，他还在做更多的尝试，发表不同寻常的激进言论。下面就是他的一些最新的观点。

- 停止关注苹果公司，它最富有创造性的时期已经过去了，不再能够改变世界，是时候寻找下一家能够“think different”的创业公司了。
- Google的竞争力已经开始衰退，它越来越像那些充满官僚主义的传统大公司，十年来第一次，竞争对手有可能做出更好的搜索引擎。
- CPU的主频增长放缓，多核CPU成为未来的方向，这意味着并行计算将大行其道，如果有人能够编写出将计算自动分配到多个CPU的软件，他将大获成功。
- 网络支付（online payment）的现状非常糟糕，有巨大的改进空间，未来几年这个领域会涌现大批创业公司。
- 电影和电视业将在未来几十年中逐渐消亡，杀死它们的不是盗版，而是更好的娱乐的方式。







西乔
设计师，项目经理。2006年起携创业团队从事Web技术外包开发及产品咨询顾问。
如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，请给西乔发邮件：arthur369@gmail.com

